Издательство «Знание»

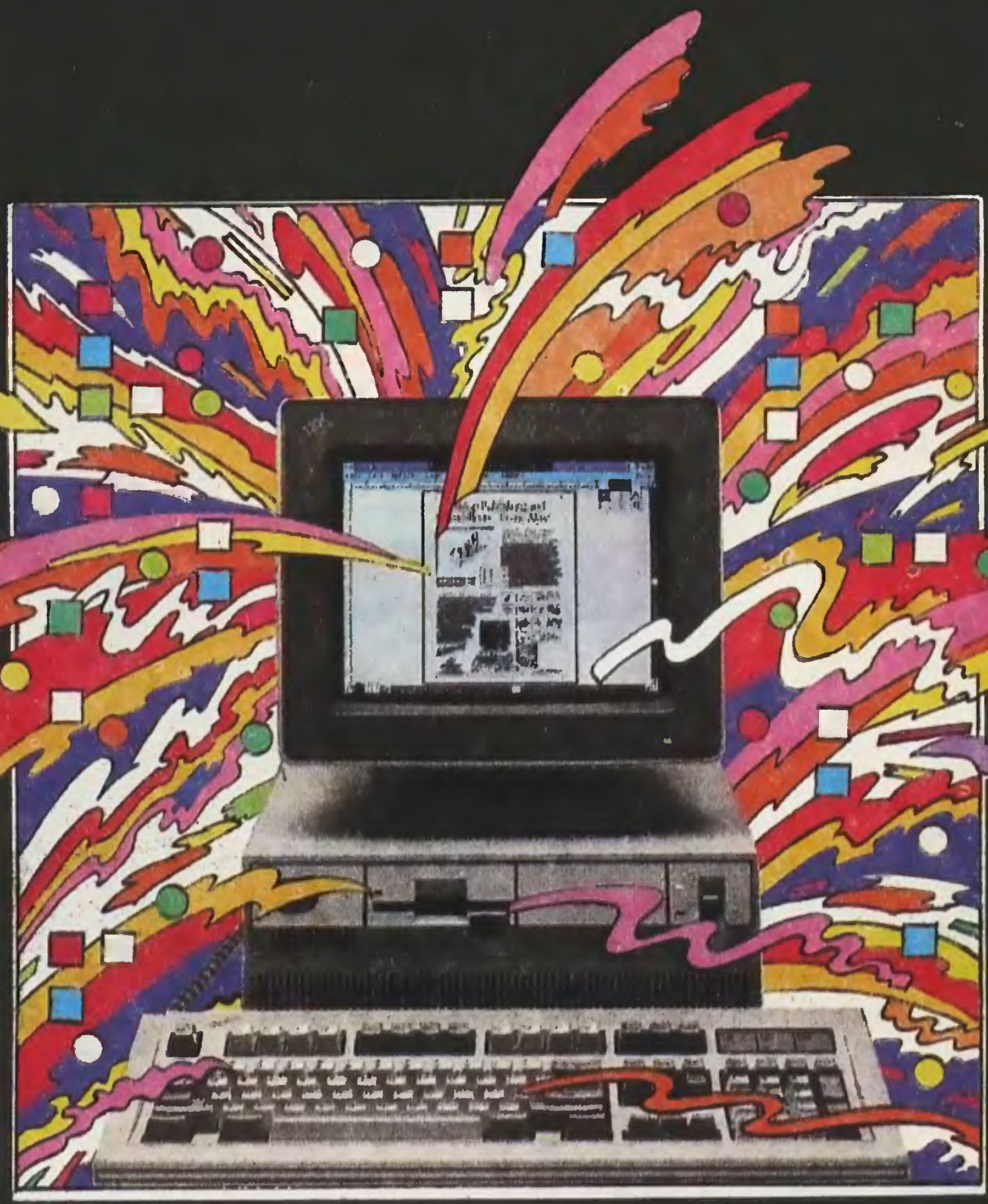
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Операционная система UNIX



1991

6

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

6/1991

Издается
ежемесячно
с 1988 г.

ОПЕРАЦИОННАЯ СИСТЕМА UNIX

В Номере:

В.В.Леонас
ИЗ ИСТОРИИ СОЗДАНИЯ ОС UNIX

В.И.Беляков
СТАНДАРТИЗАЦИЯ UNIX

М.М.Денисов, Д.И.Торопов
КОМПЬЮТЕРЫ U-СЕРИИ

А.В.Каширин, В.А.Николаев
ЯЗЫКОВЫЕ СРЕДСТВА NeWS

Питер Нортон
ВЗГЛЯД В ПРОШЛОЕ, ПЕРСПЕКТИВЫ БУДУЩЕГО

А.А.Ковалев
ТУРБО СИ ДЛЯ НАЧИНАЮЩИХ

Москва
Издательство
"Знание"
1991

ББК 32.85
О 53

Авторы выпуска

ЛЕОНАС ВЛАДАС ВЛАДАСОВИЧ - кандидат физико-математических наук, специалист в области прикладных программ.

БЕЛЯКОВ МИХАИЛ ИОСИФОВИЧ - кандидат технических наук, специалист в области компьютерной техники.

КАШИРИН АЛЕКСАНДР ВАСИЛЬЕВИЧ - научный сотрудник Ленинградского института информатики и автоматизации АН СССР, специалист в области архитектуры ЭВМ.

НИКОЛАЕВ ВЛАДИМИР АЛЕКСЕЕВИЧ - младший научный сотрудник Ленинградского института информатики и автоматизации АН СССР.

КОВАЛЕВ АНДЩРЕЙ АЛЕКСАНДРОВИЧ - инженер-программист. Профессиональные интересы - экспертные системы, базы данных и их применение в информационно-справочных системах и САПР.

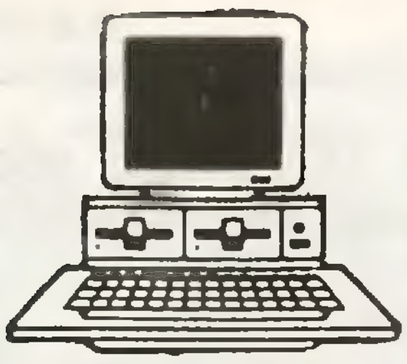
ДЕНИСОВ МИХАИЛ МИХАЙЛОВИЧ - директор по маркетингу СП "Юнимас".

ТОРОПОВ ДМИТРИЙ ИЛЬИЧ - менеджер по маркетингу СП "Юнимас".

Редактор Б. М. ВАСИЛЬЕВ

ISBN 5-07-001720-9

©Издательство "Знание", 1991 г.



Основанная на концепции монопольной однопрограммности, ОС MS DOS в настоящее время вызывает у пользователей большее неудовольствие, чем, например, недостаточно высокая вычислительная мощность компьютера.

В.В.Леонас

Из истории создания ОС UNIX

Одной из очень часто встречающихся задач, решение которой крайне затруднено при использовании ПК, работающих в монопольном однопрограммном режиме, является задача разделения информации. Такая задача возникает, например, при необходимости автоматизации учрежденческой деятельности, для чего, как правило, оказывается необходимым использование компьютерных сетей и централизованных баз данных. Наиболее удобным средством для организации централизованных баз данных в компьютерных сетях является файл-сервер. Он представляет собой узел компьютерной сети, в функции которого входят хранение больших объемов информации и обеспечение доступа к ней из остальных узлов этой сети. Компьютеры, используемые для создания на их основе файл-серверов, должны обладать достаточными информационными и вычислительными ресурсами. Наиболее просто и эффективно функции файл-сервера реализуются при использовании компьютера, работающего под управлением многопользовательской многопрограммной ОС разделения времени. В качестве такой ОС чаще всего используют ОС ONIX или подобные ей операционные системы.

Даже фирма IBM, в течение длительного времени не признававшая идеологию ОС UNIX, предлагает теперь подобные ей ОС: ОС PCIX - пользователям своих ПК типа XT и AT и ОС AIX - пользователям ПК модели 80 из семейства PS/2. Новое семейство RISC - компьютеров IBM, получившее название RS 6000, полностью ориентировано на использование ОС UNIX.

В настоящее время сочетание слов "персональный компьютер" и "операционная система (ОС) UNIX" воспринимается уже совсем не так фантастически, как лет 5-10 назад. Действительно, в те годы сообщение о разработке версий ОС UNIX для 8-разрядных персональных компьютеров (ПК) на основе микропроцессоров Intel 8080 и Z8000, а также для младших моделей 16-разрядных ПК серии LSI-11 могло вызвать в лучшем случае снисходительную усмешку. Появление версий ОС UNIX для ПК на основе микропроцессоров Intel 8086 и Intel 8088 (например, ОС VENIX) продемонстрировало лишь принципиальную возможность того, что ОС этого семейства могут работать (неважно, плохо или хорошо) на 16-разрядных ПК. Опыт раработки ПК на основе микропроцессора Intel 8088 и версий ОС UNIX для них (в первую очередь речь идет, конечно, об ОС XENIX) показал реальность использования ОС этого семейства на 16-разрядных ПК. Однако только появление компьютеров на основе микропроцессора Intel 80386, по-прежнему называемых персональными, но которые язык уже (или, может быть, еще?) не поворачивается так называть, сделало, по-видимому, сочетание слов "ПК" и "ОС UNIX" неразрывным на будущее. Этому также способствовало успешное завершение процесса стандартизации ОС UNIX (практически разработан процесс стандартов POSIX).

Возникает законный вопрос: чем обусловлена необходимость создания новых ОС для ПК, если есть такая популярная ОС, как MS-DOS, для которой разработано фантастически большое количество пакетов прикладных программ из самых разных предметных областей? Не вдаваясь в обсуждение ситуации с "виртуальной" (она как бы есть, но ее как бы и нет) операционной

системой OS-2, попытаемся ответить на поставленный вопрос.

Даже простое сравнение ОС UNIX и MS-DOS показывает, что первая из них по своим возможностям намного богаче второй. Данное обстоятельство определяется в первую очередь различной природой этих двух операционных систем; как уже говорилось, ОС UNIX является многопользовательской многопрограммной операционной системой, а MS-DOS представляет собой однопользовательскую однопрограммную операционную систему (строго говоря, ее и операционной системой назвать трудно - это, скорее, монитор). Поклонники MS-DOS, не знакомые с настоящими многопользовательскими многопрограммными ОС, обычно возражают против этого, ссылаясь в качестве примера на среду MS Windows, которая не является по-настоящему многопрограммной средой (о многопользовательской среде здесь речи вообще идти не может). Кроме того, в OS UNIX максимальная длина программ практически не ограничена - она может достигать 16 Мбайт, в то время как в MS-DOS эта величина не превышает 640 Кбайт. Наконец, такие поддерживаемые OS UNIX функциональные возможности, как электронная почта, сетевые средства, управление планированием выполнения программ, средства автоматического вызова программ на выполнение через заданные интервалы времени или в определенные моменты времени, контроль доступа к информации, различные многооконные системы (например, системы Multiview и Xsight, соответствующие стандарту X11) и другие делают эту ОС привлекательной для пользователей.

Согласно формальной классификации ОС UNIX следует отнести к категории многопользовательских многопрограммных операционных систем, работающих в режиме разделения времени. Однако гибкость ОС UNIX обусловила удобство ее адаптации для работы как в режиме реального времени (примером такой адаптации может служить ОС UNOS, разработанная фирмой Charles River Data Systems), так и в режиме пакетной обработки (примерами могут служить различные версии ОС UNIX для суперкомпьютеров) или даже при различных сочетаниях всех трех режимов (например, ОС UNOS обеспечивает возможность одновременной работы как в режиме реального времени, так и в режиме разделения времени).

Следует отметить и такую особенность ОС UNIX, как мобильность, что позволяет легко переносить ее на компьютеры с различной архитектурой. Простота и удобство всех уровней интерфейса с пользователем (в данном случае пользователем является программист) обеспечивают прозрачность и ясность программ, создаваемых в среде ОС UNIX. Более того, широчайший набор имеющихся в среде ОС UNIX инструментальных средств позволяет с их помощью решать широкий класс задач, вообще не прибегая к традиционному программированию.

Распространение ОС UNIX на мир персональных компьютеров в значительной степени обусловлено недостатками ОС MS-DOS, связанными с ее однопользовательской однопрограммной природой. Этому способствуют два обстоятельства. Во-первых, крупнейшие фирмы - разработчики пакетов прикладных программ (например, Ashton-Tate, Lotus и Samna International) перенесли свои наиболее популярные пакеты из среды OS MS-DOS в среду ОС UNIX (точнее говоря, в среду ОС XENIX). Во-вторых, в среде различных версий ОС UNIX, предназначенных для работы на персональных компьютерах, имеются средства, обеспечивающие возможность выполнения программ, разработанных для среды ОС MS-DOS (к таким средствам относится, например, пакет VP/ix для ОС XENIX и ОС DOS-Merge).

Следствием огромного интереса к ОС UNIX со стороны столь большого числа крупнейших фирм-разработчиков аппаратных и программных средств вычислительной техники явилось появление значительного числа не полностью совместимых друг с другом версий ОС UNIX. Уже в 1985 г. число таких версий перевалило за 70. Наличие столь большого числа не полностью совместимых друг с другом версий ОС UNIX вызвало серьезную озабоченность среди пользователей, которые по вполне понятным причинам хотят иметь единую унифицированную операционную среду, что невозможно без стандартизации ОС UNIX.

Большую работу по стандартизации ОС UNIX проводит основанная в конце 1984 г. Ассоциация производителей аппаратных и программных средств вычислительной техники X/OPEN. Весной 1985 г. эта Ассоциация подготовила документ X/OPEN Portability Guide, а к лету 1986 г. было продано уже свыше 7000 его копий. Второе, расширенное и дополненное издание этого документа появилось в 1987 г.

В 1988-1989 гг. к решению проблемы стандартизации ОС UNIX присоединилось еще два участника: косорциум Open Software Foundation (OSF), созданный по инициативе фирм IBM и DEC, и коксорциум UNIX International, организованный в противовес OSF по инициативе корпорации AT&T и фирмы Sun Microsystems. Появление этих двух новых участников внесло дополнительные трудности в процесс стандартизации, так как ознаменовало собой возникновение двух противостоящих группировок. Одну из них возглавляет фирма IBM, которой не дает покоя лидерство корпорации AT&T в вопросах, связанных с ОС UNIX, а вторую - корпорация AT&T, не желающая это лидерство уступать.

Говоря о стандартизации ОС UNIX, нельзя не упомянуть об успешно завершившейся стандартизации языка Си, для которого уже принят стандарт Американского национального института стандартов (ANS1).

Переходя к обсуждению перспектив развития ОС UNIX, необходимо отметить ряд обстоятельств. Во-первых, консорциум OSF объявил о разработке собственной версии ОС UNIX, которая получит название OSF-1. Во-вторых, корпорация AT&T, предполагающая в дальнейшем вместо языка Си использовать язык Си++, заявила о том, что новая версия System V Release 5 будет реализована на языке Си++ (этот язык, разработанный Б.Страуструпом, сотрудником фирмы Bell Laboratories, представляет собой дальнейшее развитие языка Си; язык Си++ позволяет использовать объектно-ориентированный подход к программированию и абстрактные типы данных). В-третьих, по оценке американского журнала "Electronics", доли рынка, приходящиеся на ОС UNIX и MS-DOS, к концу 1991 г. сравняются и каждая из них оритентировочно составит 22-23% от всего объема продаж.

Во второй половине 60-х гг. фирма Bell Laboratories была вовлечена в работы по проекту Multics, в которых принимали также участие фирма General Electric и Массачусетский технологический институт. В цели проекта Multics входило создание многопользовательской операционной системы разделения времени.

В 1969 г. фирма Bell Laboratories вышла из проекта Multics и полностью прекратила все связанные с этим проектом работы. Наиболее активное участие в проекте Multics со стороны фирмы Bell Laboratories принимали К.Томпсон, Д.М.Ритчи, М.Д.Макилрой, Дж.Ф. Осанна. Им не хотелось терять накопленные наработки, поэтому, несмотря на запрет руководства продолжать работы по созданию ОС, они начали искать различные способы использования полученных ими результатов. Например, ими рассматривались такие варианты, как разработка новой ОС для имевшегося в фирме Bell Laboratories компьютера GE-645 (для которого создавалась ОС Multics) и приобретение компьютера среднего класса (например, PDP-10 или Sigma-7) с целью разработки новой ОС для него. Параллельно с этим на имевшемся компьютере GE-645 осуществлялось моделирование файловой системы и схем управления памятью. Однако в силу ряда обстоятельств обе эти идеи были отвергнуты.

В это время К.Томпсон занимался разработкой программы Space Travel, моделировавшей движение основных небесных тел, входящих в состав Солнечной системы. Каждый прогон этой программы на компьютере GE-645 обходился в 75 долл., что было непомерно дорого. Поэтому К.Томпсон принялся за поиски какого-либо другого компьютера, использование которого обходилось бы не так дорого. В ходе своих поисков он обнаружил практически никем не используемый мини-компьютер PDP-7 — предшественник широко известных мини-компьютеров семейства PDP-11.

Тогда К.Томпсон начал перерабатывать свою программу Space Travel с тем, чтобы ее можно было использовать на мини-компьютере PDP-7. Однако оказалось, что переход к использованию мини-компьютера PDP-7 не так уж

прост, поскольку имевшееся для него программное обеспечение состояло лишь из ассемблера и компоновщика. При этом необходимо было использовать перфоленту. Да и работать на этом мини-компьютере можно было лишь в монопольном режиме. Отсутствие программных средств, обеспечивающих удобство разработки, отладки и выполнения программ, натолкнуло К.Томпсона на мысль о создании соответствующих программных средств. Так на новом витке возникла идея разработки собственной операционной системы.

В 1969 г. появилась написанная на языке АССЕМБЛЕР однопользовательская ОС для мини-компьютера PDP-7, перенесенная через некоторое время на мини-компьютер PDP-9 (являвшийся также предшественником широко известного семейства мини-компьютеров PDP-11). Именно эта ОС и получила впоследствии название UNIX. Это название было предложено Б.У.Керниганом в 1970 г. и отражало противопоставление ОС Multics (простая и маленькая ОС UNIX — сложная и большая ОС Multics), ОС UNIX заимствовала определенные черты у ОС Multics.

В 1970 г. по предложению Дж.Ф.Осанны фирма Bell Laboratories приобрела мини-компьютер PDP-11/20, который предполагалось использовать в проекте, связанном с разработкой системы, облегчающей создание документации. В 1971 г. ОС UNIX была перенесена на мини-компьютер PDP-11/20, в качестве основы для системы ведения документации. В июне 1972 г. появилась новая версия ОС UNIX, также написанная на языке АССЕМБЛЕР и известная впоследствии как версия 2.

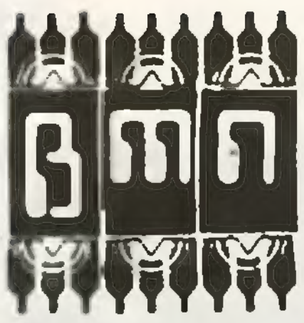
В это время К.Томпсон уже работал над созданием безтипового языка Би, строившегося на основе также безтипового языка Би-Си-Пи-Эл. В результате был создан новый язык — Эн-Би. Наконец занимавшийся вопросами генерации кода для транслятора с языка Эк-Би Д.М.Ритчи создал на его основе получивший впоследствии широчайшую известность язык Си.

В 1973 г. впервые появилась версия 3. В мае 1975 г. создана первая пригодная для промышленной эксплуатации версия ОС UNIX, известная в настоящее время как версия 6. В декабре 1977 г. ОС UNIX была перенесена на компьютер Interdata 8/32. В результате такого переноса сформировалась еще одна версия ОС UNIX, в настоящее время известная как версия 7. В 1978 г. ОС UNIX была перенесена на микрокомпьютеры семейства VAX-11, эта версия получила название 32V.

После создания ОС UNIX в течение длительного времени она была известна лишь в узком кругу специалистов по операционным системам, поскольку использовалась лишь в фирме Bell Laboratories и вычислительных центрах ряда американских университетов. Затем последовал период многочисленных переносов ОС UNIX на компьютеры различных типов и создания ОС, подобных ОС UNIX. В истории создания и развития ОС UNIX особое место занимает 1980 г., когда ОС UNIX и подобные ей системы начали активно использоваться, в том числе и на персональных компьютерах. Именно в 1980 г. ОС UNIX привлекла к себе внимание делового мира — основного пользователя персональных компьютеров.

В этом году корпорация AT&T (в которую входит фирма Bell Laboratories) выпустила версию ОС UNIX, получившую название System III, а в январе 1983 г. — версию System V. В январе 1984 г. корпорация AT&T выпустила версию ОС UNIX, получившую название System V Release 2.0, претендующую на то, чтобы стать стандартом.

Разработка ОС UNIX была высоко оценена специалистами. В 1982 г. К.Томпсон и Д.М.Ритчи за создание ОС UNIX были удостоены "Премии за достижения", ежегодно присуждаемой редакцией журнала Electronics. При этом необходимо отметить, что впервые за 9 лет с момента ее учреждения эта премия была вручена за разработку программного обеспечения, а не за разработку аппаратных средств. В 1983 г. за создание ОС UNIX R.Томпсон и Д.М.Ритчи были удостоены сразу двух премий Ассоциации специалистов по вычислительной технике: премии Тьюринга и премии за разработку нового программного обеспечения.



Весной 1985 г. корпорация AT&T опубликовала документ, названный ею System V Interfage Definition (SVID), представляющий собой стандарт на ОС UNIX версии System V, а в 1986 г. — расширенное издание этого документа в трех томах. Деятельность по стандартизации ОС UNIX в настоящее время осуществляется рабочей группой P 1003 IEEE, разработавшей стандарт Posix на мобильную ОС. Первый проект стандарта был подготовлен этой рабочей группой еще весной 1986 г.

Б.И.Беляков

Стандартизация UNIX

Широкое распространение, несомненные достоинства и хорошие перспективы операционной системы UNIX были начиная с 1984 г. поддержаны новым важным шагом — попытками стандартизовать ее пользовательские интерфейсы и язык Си, на котором система в основном написана. Существенный прогресс к настоящему моменту достигнут в этом направлении следующими организациями:

- комитетом ANS1 X3L11 Американского национального института стандартов, разработавшим стандарт языка Си с учетом системных библиотек;

- комитетом IEEE P1003, разрабатывающим стандарт POSIX на пользовательские интересы системы;

- группой X/Open, объединившей с целью выработки такого стандарта ряд западноевропейских, американских и японских компаний.

Американский национальный стандарт языка Си разработан Техническим комитетом X3J11 и адресован как программистам, использующим Си, так и создателям конкретных реализаций языка. Цель разработки стандарта состоит в том, чтобы дать недвусмысленное и машинно-независимое описание языка Си. Поскольку в языке нет изобразительных средств для ввода-вывода и многих других типовых операций, стандарт включает достаточно содержательную библиотеку функций как неотъемлемую часть языка. В библиотеке, в частности, имеются средства локализации, т.е. учета национального представления различных величин и обозначений.

Проект стандарта интерфейсов мобильной операционной среды, названный POSIX, подготовлен рабочими группами Американского института IEEE. Название мобильной системы — POSIX — является товарным знаком IEEE. В стандарте воплощена точка зрения на чисто мобильный интерфейс программ с операционной системой.

К настоящему моменту выпущена окончательная версия проекта стандарта POSIX на общие функции программного интерфейса. Другие документы POSIX еще требуют доработки. Программный интерфейс POSIX реализуется как в операционных системах, совместимых с UNIX, так и в системах, построенных на других концепциях.

В состав материалов POSIX входит общее руководство по стандартам POSIX, в частности описание системной архитектуры POSIX. Наиболее проработан общий программный интерфейс POSIX. От традиционных вещей общего программного интерфейса UNIX документ отличается специальными функциями обработки сигналов, специальными функциями работы с терминалом и рядом мелких особенностей.

Основой командного языка в POSIX является язык интерпретатора Bourne shell — старейший из командных интерпретаторов UNIX. Сделаны расширения в духе двух других наиболее известных интерпретаторов C shell и Korn shell. Введена возможность редактирования текущей командной строки или одной из предыдущих командных строк в стиле одного из текстовых редакторов UNIX. В командный интерфейс входит система извлечения прикладных программ из дистрибутивного носителя, установки их в требуемых местах системы, поддержки базы данных, доступной пользователям и

прикладным программам и используемой для отображения логических ресурсов программы на реальные ресурсы системы.

В материалах POSIX сделана начальная проработка интерфейса реального времени. Имеются две части этого стандарта: часть "a", посвященная понятию параллелизма внутри процесса (Thread), и часть "b", посвященная расширениям для целей реального времени. Реальное время в операционных системах определено в стандарте как способность операционной системы обеспечить требуемый уровень сервиса в условиях, когда время реакции ограничено. Стандарт предусматривает многие функциональные возможности для реализации реального времени.

1. Двоичные семафоры — минимум примитивов синхронизации как основа для построения более сложных средств синхронизации в прикладных программах; семафор является новым типом специального файла; операции: установить, ждать, ждать условно.

2. Блокировка процессов в памяти — для устранения системных издержек при считывании процесса из памяти второго уровня.

3. Разделяемая память — одновременный доступ прикладных программ к общему участку их образов памяти; разделяемая память выступает как еще один тип специального файла. Допустимы общие операции работы с файлами типа открытия-закрытия, а кроме того, вводятся специальные процедуры, связанные отображением разделяемой области в адресное пространство процесса. Ряд типовых файловых функций из программного интерфейса POSIX (смена владельца, кода защиты, работа с именами связей и т.п.) допустим для специального файла разделяемой памяти.

4. Приоритетное планирование:

- допускается до 32 приоритетов, для каждого приоритета свой список процессов;
- для каждой дисциплины планирования свой диапазон приоритетов, который может вырождаться либо до одного приоритета, либо до полного диапазона приоритетов; дисциплина означает определенные правила включения в список и упорядочения списка;
- независимо от дисциплины при выборе нового процесса для занятия процессора берется процесс из головы списка с наибольшим приоритетом.

5. Допустимы три дисциплины: FIFO (в порядке активизации), round robin (FIFO с квантованием времени) и по крайней мере еще одна дисциплина; в случае FIFO список упорядочивается по времени активизации процессов; в случае квантования список упорядочивается, как при FIFO, но есть дополнительная причина для снятия процесса с процессора (исчерпание кванта); стандарт требует, чтобы реализация включала еще хотя бы одну дополнительную дисциплину.

6. Асинхронные прерывания — средства, заменяющие механизм сигналов, не совместимые с ним и позволяющие организовывать очередь асинхронных прерываний.

7. Таймеры — два типа таймеров: срабатывающий один раз и срабатывающий периодически; задаваемое время может быть абсолютным и относительным (к текущему моменту).

8. Межпроцессное (внесетевое) взаимодействие.

9. Синхронный ввод-вывод — более детальные (чем имеющиеся сейчас) процедуры синхронного ввода-вывода, обеспечивающие контроль целостности данных и файлов со стороны прикладной программы.

10. Асинхронный ввод-вывод.

11. Файлы реального времени.

Пока в проекте стандарта для расширения реального времени описаны только семафоры, разделяемая память, приоритетное планирование, таймеры и синхронный ввод-вывод.

Понятие "thread" является новым для UNIX и обозначает единицу последовательного выполнения внутри процесса. Назовем его подпроцессом, хотя вполне возможны другие переводы. Каждый подпроцесс имеет свой идентификатор (в пределах процесса), приоритет и дисциплину планирования, таймеры, код ошибки (переменную errno), системные ресурсы, необходимые для функционирования. Подпроцессы функционируют внутри единого адресного пространства процесса.

Подпроцессы могут быть использованы для параллелизма, присущего многооконному интерфейсу, обработке событий в режиме реального времени, задачам языка АДА, обработке транзакций, сетевым и распределенным систе-

мам. Процессы взаимодействуют между собой независимо от их внутреннего строения. Реализован ли процесс в виде единой программы или является совокупностью подпроцессов — этот факт не влияет на окружение процесса. Подпроцессы прежде всего нужны там, где цена создания нового параллельного действия соизмерима с ценой выполнения вызова функции. Поэтому подпроцессы должны создаваться максимально быстро.

В стандарте рассматриваются:

- средства синхронизации подпроцессов, обеспечивающие взаимное исключение; используются специальные синхронизирующие объекты, называемые mutex (mutual exclusion), и условные переменные; механизм синхронизации напоминает исключение с помощью критических участков;
- различные способы завершения подпроцесса;
- работа традиционных функций управления процессами и сигналами в случае подпроцессной структуры процессов;
- проблемы повторной входимости в случае подпроцессной структуры процессов.

Интерфейс защиты (управляемый доступ) разработан в POSIX достаточно глубоко и резко отличается от традиционного подхода UNIX к этой проблеме. Объекты, на которые распространяется управляемый доступ, — файлы всех типов (включая программные каналы) и процессы. Файлы выступают в роли объектов, процессы — в роли субъектов. Различается доступ к объектам, основанный на дискретном выборе, и доступ, основанный на полномочиях.

Дискретный выбор подразумевает соответствие списков, когда для каждого объекта есть список субъектов, доступ которых к объекту разрешен. Таковым является механизм доступа в имеющихся версиях UNIX, и он достаточен для многих применений.

Доступ, основанный на полномочиях, использует соответствие меток. Вводятся метки объектов и субъектов, т.е. процессов и файлов. Вводятся понятия доминанты и равенства меток (как отношения между метками). Создаваемый файл наследует метку от создавшего его процесса. Вводятся соотношения, определяющие права процессов по отношению к файлам.

Несмотря на то что дискретный доступ соответствует по философии механизму защиты, принятому в UNIX, интерфейс его резко отличается от используемого. Вводится существенная детализация механизма и большое количество новых понятий, функций и команд, реализующих этот механизм. Вводимые средства можно разделить на такие группы:

- для работы со списками доступа при дискретной защите;
- для проверки права доступа;
- для управления защитой на основе полномочий;
- связанные с привилегиями.

В рамках POSIX начались работы над интерфейсом системного администратора. Работы предполагается завершить к 1993 г. К проблеме системного администратора в POSIX подходят с объектно-ориентированных позиций. Предполагается определить объекты, имеющие интерес с точки зрения системы, удовлетворяющей спецификациям POSIX, а затем определить множества действий, которые можно выполнять над такими объектами. В качестве классов предполагаются пользователь, группа пользователей, устройство, файловая система, процесс, очередь, вход в очередь, машина, система, администратор, программное обеспечение и пр. Определяются атрибуты таких классов, операции над классами и события, которые могут с ними происходить.

Такой подход, безусловно, изящный с точки зрения теории, по-видимому, не просто сделать удобным на практике, учитывая, что работа системного администратора уже имеет свои наработанные годами традиции. Впрочем, у рабочей группы есть еще довольно много времени для реализации своих идей, так что пока рано говорить о том, каков будет интерфейс системного администратора.

Сетевым интерфейсам посвящен проект стандарта, состоящий из нескольких частей. Одна часть стандарта характеризует доступ к файлам в сетевой среде и состоит из трех секций: секция, описывающая программный интерфейс независимо от языка программирования; секция, зависящая от используемого языка программирования; секция, описывающая командный интерфейс доступа к файлам. Первая секция получена извлечением из P1003.1 всего того, что относится к доступу к файлам. Затем поведение

функций в нераспределенной среде дополняется описанием их поведения в сетевом варианте.

Вторая часть сетевого стандарта посвящена межпроцессному взаимодействию в сетевой среде. Взаимодействие процессов в сетевой среде рассматривается как расширение программного интерфейса POSIX. Рассматриваются следующие вопросы:

- установка и разрыв соединения;
- передача данных при межпроцессном взаимодействии;
- управление событиями;
- функционирование сети;
- средства, не зависящие от сетевого протокола;
- средства, зависящие от сетевого протокола.

В рамках POSIX рассматривается связь с программами на ФОРТРАНе 77. Прежде всего дается фортранное определение констант и макросов, используемых в программном интерфейсе POSIX. Поскольку этот интерфейс основан на структурах данных языка Си, возникает проблема доступа к ним из фортранных программ. Решение сделано в духе абстрактных типов данных, представляющих требуемые агрегатные структуры языка Си. Вводятся специальные фортранные подпрограммы (subrotine) для манипуляции с такими структурами. Дается фортранное изображение ряда примитивных функций из общего программного интерфейса POSIX: fork, exec, wait и т.п.

Целью проекта X/Open является создание среды прикладного программирования (common applications environment — CAE) для функционирования мобильных прикладных программ, которые без модификаций или с минимальными модификациями могут быть перенесены с одной машины на другую. Основная философия стандартизации в рамках X/Open: не создавать новые стандарты, а использовать имеющиеся. В этом основное отличие проекта X/Open от других проектов и причина быстрого продвижения проекта по степени охвата различных интерфейсов, включая интерфейсы с языками программирования, средства управления данными, межпроцессное и сетевое взаимодействие. Однако этот факт определяет несамостоятельность группы и ее зависимость от фактического и юридического статуса других проектов.

Первоначально предложения группы X/Open основывались на спецификации фирмы AT&T для разработанной ею UNIX System V и известных под названием SVID (System V Interface Definition). Выпущенная в декабре 1988 г. третья версия руководства опирается на стандарт языка Си и программный интерфейс POSIX. Проблемы интернационализации разработаны в предложениях X/Open достаточно глубоко. Поддержка национального языка подразумевает:

- 8-битное кодирование символов;
- раздельное существование программы и выдаваемых ею сообщений (сообщения переводятся на требуемый язык и извлекаются во время работы программы);
- механизм объявления национального языка и национальных обозначений;
- специальная библиотека функций, ориентированных на поддержку национальных языков, в том числе динамическое определение специфических для данного языка данных;
- регулярные выражения, ориентированные на поддержку национальных языков;
- специальные команды, ориентированные на поддержку национальных языков.

Описание КОБОЛа опирается на ANSI стандарт КОБОЛа 1985 г. и его международный эквивалент. Описание ФОРТРАНа опирается на ANSI стандарт 1978 г., описание ПАСКАЛЯ — на международный стандарт 1983 г., описание языка запросов SOL — на ANSI стандарт 1986 г., описание последовательного метода доступа ISAM — на описание фирмы Informix Corporation. Многооконный интерфейс соответствует X Window. Механизм межпроцессного взаимодействия в предложениях X/Open включает в себя семафоры, разделяемую память и посылку сообщений. Интерфейс семафоров и разделяемой памяти отличен от этих механизмов в POSIX и соответствует их реализации в UNIX System V.

Стандартизация интерфейсов операционной системы является беспрецедентным шагом, и появление юридического стандарта поднимет, как ожидается, и без того высокие акции UNIX.

М.М.Денисов, Д.И.Торопов

Компьютеры U-серии

В последнее время в мире возрастает интерес к операционной системе (ОС) Unix. Открытая реализация Unix и OSI-возможностей используются для взаимодействия и обмена информацией между системами различных поставщиков. Ряд международных организаций, в частности Unix International, работают в области внедрения Unix - систем и стандартов на ОС Unix.

Компьютеры U - серии, производства корпорации Unisys, являющейся крупнейшим в Европе поставщиком машин этого типа, при реализации своих систем стремится к возможно большей их преемственности, открытости и соответствии стандартам. Unisys в значительной степени основывается на концепцию Взаимодействия Открытых Систем и стандарт на ОС Unix.

Автоматизация деятельности небольших подразделений требует, как правило, индивидуальных решений, которые должны без больших дополнительных затрат адаптироваться к постоянно изменяющимся требованиям жизни. Секретариаты, плановые отделы, бухгалтерии, отделы сбыта и снабжения, отделы разработчиков, программистов и т.д. нуждаются в дополнительных вычислительных системах, не требующих специальных помещений, и способных удовлетворить разнообразные коллективные и индивидуальные требования. Обмен данными с ЭВМ различных служб и уровней, способность к расширению и модификации также необходимы для перспективных систем.

Unix - ориентированные компьютеры U-серии образуют возрастающий по мощности ряд ЭВМ от настольных рабочих станций до многопроцессорных супермини, обладающих широкими коммуникационными и сетевыми возможностями для связи с удаленными терминалами, рабочими станциями и с HOST - компьютерами.

Стандартизованная версия операционной системы Unix - Unisys System V делает эти ЭВМ совместимыми по программному обеспечению между собой и с другими UNIX - ориентированными системами.

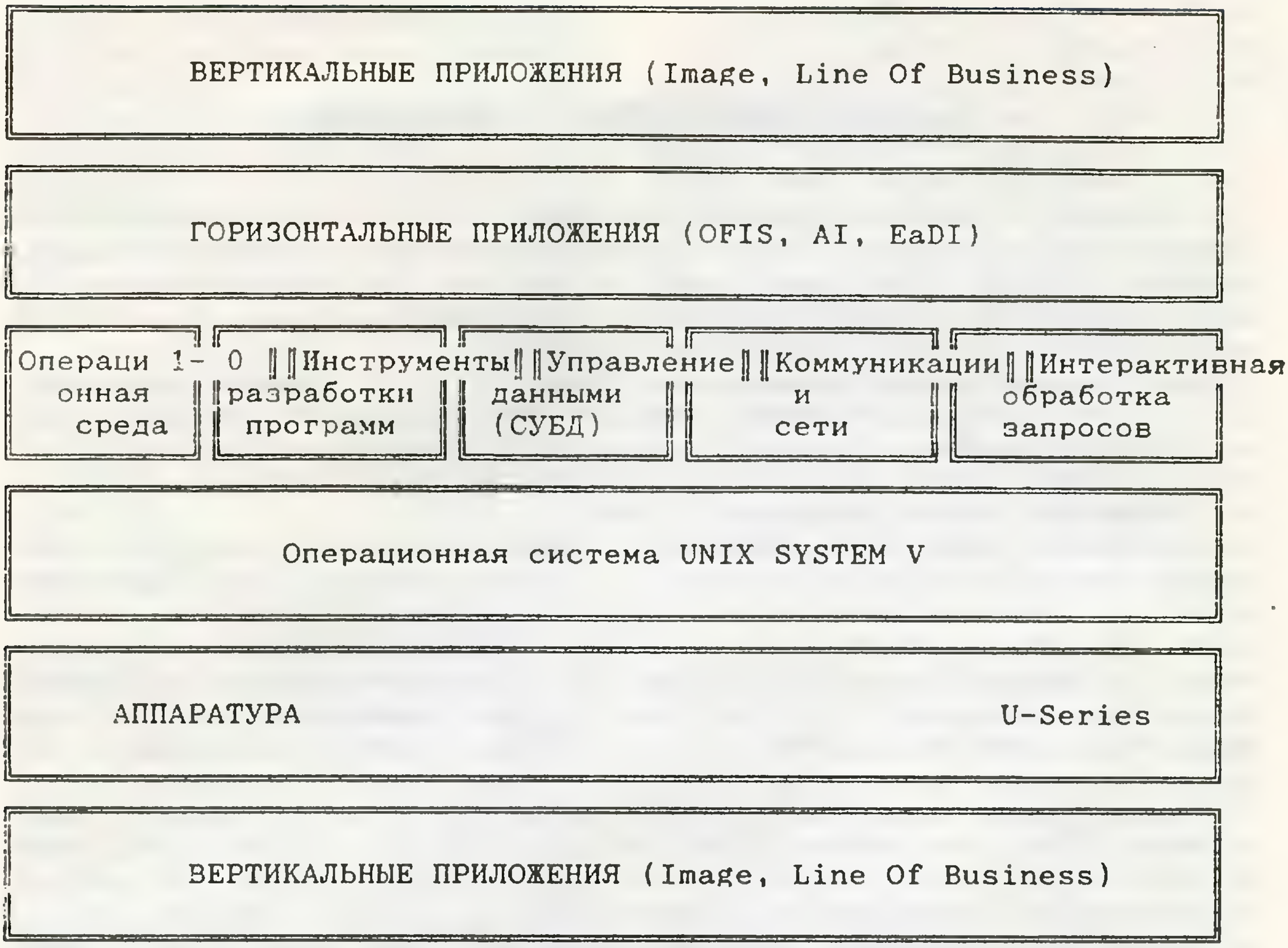
Широкий спектр возрастающих по мощности компьютеров U-Серии и уникальный набор программного обеспечения дают возможность Unisys построить расширенную платформу VAR.

Value Added Tkatform (VAP)

Понятие value added широко используется в англоязычной литературе. Value added означает, что к исходному продукту добавляется нечто, что может быть расширением возможностей применения исходного продукта или некий набор услуг или что-то другое, что увеличивает потребительскую ценность исходного продукта. Далее мы будем употреблять словосочетание "расширенная платформа".

Расширенная платформа VAR - это концепция, включающая аппаратуру, программное обеспечение и обслуживание пользователей, разработанная для обеспечения открытыми UNIX-системами как больших, так и малых пользователей. В этой концепции Unisys объединяет свои обязательства по поддержке открытых систем и лидерство в области разработки UNIX-продукции. Компоненты Value Added Platform могут быть разделены по пяти уровням. Как показано на рисунке, все компьютеры U-Серии работают под управлением ОС UNIX System V (или Unisys System V), которая поддерживает ряд системных программных продуктов, представленных в виде пяти ключевых направлений использования вычислительных мощностей.

Над системами ПО расположены разработанные Unisys горизонтальные приложения, которые могут использоваться для объединения средств системного уровня между собой и со следующим уровнем вертикальных приложений. Вертикальные приложения могут быть выбраны потребителем из множества специализированных программных пакетов (Unisys или других поставщиков), работающих в среде UNIX System V.



Производимые в настоящее время UNIX-компьютеры Unisys обладают разнообразными характеристиками по объемам памяти и производительности, что дает пользователю возможность выбора подходящей моодификации.

Над системным ПО расположены разработанные Unisys горизонтальные приложения, которые могут использоваться для объединения средств системного уровня между собой и со следующим уровнем вертикальных приложений. Вертикальные приложения могут быть выбраны потребителем из множества специализированных программных пакетов (Unisys или других поставщиков), работающих в среде UNIX System V.

Все ЭВМ U-Серии работают в обычных климатических условиях рабочего помещения (офиса) и не требуют специального сетевого питания и специальных условий по температуре и влажности.

Операционная система Unisys System V

Unisys System Operating System - это полностью сертифицированный продукт, развитие стандартной операционной системы UNIX System V Release 3.2 фирмы AT&T. Unisys System V включает все стандартные возможности UNIX, совместима с промышленными стандартами:

- FIPS 151.1 NIST;
- 1EEE 1003.1 POSIX;
- X/Open XPG2,

а также расширена за счет включения таких средств, как:

- встроенная поддержка разнообразных национальных языков MNLS (Multi-National Language Supplement);
- транспортный интерфейс TLI (Transport Layer Interfase) в соответствии с моделью OSI/ISO;

- сетевая файловая система NES (Network File System) обеспечивающая прозрачный сетевой доступ к файлам, расположенным на различных компьютерах с различными операционными системами;
- меню-ориентированные пользовательские интерфейсы с операционной системой и утилитами;
- средства поддержки программ реального времени;
- поддержка до 4ГБ виртуального адресного пространства на одну задачу.

Программы операционной среды реализуют административные системные функции и включают:
- меню-доступ к системным функциям SA Menus;
- средства авторизации доступа Security Tools;
- средства поддержки национальных языков MNLS;
- средства оконной работы для одновременного обзора данных от разных источников MINDOWS.

U - Серия - платформа для рывка в будущее

Множество приложений, обслуживающих специфические области деятельности, уже существуют в виде типовых адаптивных пакетов. Их поставляет как Unisys, так и ряд других независимых программистских фирм. Тем не менее, конечный пользователь без особого труда с помощью мощных средств автоматизации программирования, таких, как 4GL, может самостоятельно, или пользуясь консультациями специалистов, разработать собственный пакет. Существующие программные средства включают:
- видсообработку документов;
- управление производством;
- обработку штрих-кодов;
- управление финансами;
- библиотечное дело и т.д.

Следует отметить, что библиотека программных средств Unisys для горизонтальных и вертикальных приложений исключительно богата. Организациям, которые заботятся о рентабельности и быстром вводе в эксплуатацию прикладных систем, значительно выгоднее использовать уже готовые пакетьы, чем разрабатывать с нуля новые.

Unisys с помощью Расширенной Платформы VAP обеспечивает пользователей существенными преимуществами открытых систем и сетей: - широким спектром вычислителей; - множеством системных программных компонент; - гарантиями и удобством единственного поставщика; - сервисом и поддержкой по всему миру; - взаимозаменяемостью аппаратуры и переносимостью программ; - своевременными улучшениями и расширениями.

Выбирая компьютер U-Серии, вы приобретаете на многие годы гибкий мощный инструмент увеличения производительности и прибыльности своего дела.

ДЛЯ СРАВНЕНИЯ

OS/2

- 32-разрядная система
- многозадачная система
- быстрый перенос прикладных программ DOS

DOS

- недорогая система
- есть много прикладных программ
- сложившийся рынок

ОС UNIX

- 32-разрядная система
- независимость от вида базовой машины
- многозадачная, многопользовательская система

Для создания прикладных систем с графическими пользовательскими интерфейсами используют различные программно-аппаратные платформы [2]. Одной из них является оконная система NeWS фирмы Sun Microsystems на Unix-рабочих станциях.

А.В.Каширин,
В.А.Николаев

Языковые средства NeWS

Введение

В ряду подобных систем (SunWindows, XWindow) появление NeWS было достаточно ярким событием, которое до сих пор фактически не отражено в отечественной литературе. Более того, как показала практика, при необходимости непросто получить содержательную информацию и из доступных зарубежных изданий.

NeWS представляет интерес не только как средство решения текущих практических задач. Авторы этой системы в первую очередь рассматривали ее как попытку утвердить новый перспективный подход к разработке графических интерфейсов. Способы, которыми здесь было достигнуто сочетание графики, переносимости (имеются реализации NeWS для OS/2 и Macintosh [17]), независимости от характеристик устройств ввода и отображения, сетевого сервиса, несомненно, представляют интерес для исследователей и разработчиков.

В этой статье дано общее представление о структуре и функционировании NeWS. Основное внимание сосредоточено на характеристике языковых средств разработки прикладных программ с графическими интерфейсами на базе NeWS.

Наряду с небольшой частью доступной документации по NeWS фирмы Sun Microsystems [1] основным источником информации для нас было изучение демонстрационных приложений и опыт практического применения реализации NeWS на персональной рабочей станции Personal Iris фирмы Silicon Graphics [18]. Тем не менее в списке литературы дан перечень ссылок на документы, требуемые для практического применения NeWS [3,13-16].

NeWS — торговая марка Sun Microsystems, Inc.; PostScript, Display PostScript — торговые марки Adobe Systems, Inc.; UNIX — зарегистрированный торговый знак AT&T; X Window System — торговая марка Massachusetts Institute of Technology; Personal IRIS — торговая марка Silicon Graphics, Inc.; NeXT — торговая марка NeXT, Inc.

Особенности NeWS-подхода к созданию графических интерфейсов

Дословно NeWS расшифровывается как сетевая расширяемая оконная система (Network Extensible Window System). Аналогично системе X Window [19] архитектура NeWS основана на модели "клиент-сервер". Клиенты соответствуют прикладной части сетевого ПО, а сервер — системной части, инвариантной к приложениям, предоставляющей клиентам оконный сервис. Клиенты и сервер реализуются отдельными процессами операционной системы. Взаимодействие клиента с сервером осуществляется путем передачи сообщений через коммуникационную среду.

Особенностью NeWS является механизм расширения функций сервера, на основе которого строится все взаимодействие клиентов с NeWS-сервером.

Несмотря на присутствие слова "оконный" в названии системы, на уровне базовых функций NeWS нет какой-либо фиксации стиля пользовательских интерфейсов, конструкции окон, меню и прочих элементов, которые видит на экране и которыми манипулирует пользователь. NeWS-сервер имеет примитивные

механизмы для их реализации. Настройка же сервера на обеспечение требуемого в приложении развитого оконного сервиса выполняется самим клиентом путем расширения базовых функций сервера. Пользоваться этими расширениями могут и другие клиенты.

Основную идею NeWS можно выразить следующим образом. Если в других оконных системах клиент взаимодействует с сервером через ограниченный набор типов сообщений, то каждое сообщение клиента NeWS-серверу — это программа на алгоритмически полном интерпретируемом языке программирования. Со стороны клиента NeWS-сервер выглядит как интерпретатор этого языка. При наличии в языке средств определения новых понятий сервер автоматически становится расширяемой системой. Если же в ядро языка будут встроены примитивы, достаточные для описания конструкций и алгоритмов функционирования оконных графических интерфейсов, то мы получим расширяемую оконную систему.

Таким образом, базовые функции NeWS-сервера, возможности их расширения, гибкость пользовательских интерфейсов на базе NeWS определяются выбором языка общения клиентов с сервером. В качестве этого языка был взят язык PostScript фирмы Adobe Systems [3,4,16], ядро которого было расширено дополнительными возможностями (в данной работе этот вариант PostScript будет называться NeWS PostScript).

Показанный способ развития модели "клиент-сервер" имеет прямое влияние на эффективность удаленного доступа к серверу. При решении многих задач за счет такой стратегии существенно уменьшается интенсивность взаимодействий клиента и сервера. Например, если требуется нарисовать координатную сетку на экране, то нет необходимости в цикле передавать множество линий в оконную систему. Достаточно один раз передать программу генерации полного изображения сетки.

Можно привести много примеров из классов задач, требующих возможности визуального оживления, например изменение формы некоторого графического объекта по заранее известному закону ее зависимости от текущего положения курсора. В данном случае клиент, находящийся в любой точке сети, избавляется от необходимости отслеживать движение курсора, поскольку эта функция возлагается на сервер.

Возможность одновременного обслуживания нескольких клиентов обеспечивается в NeWS-сервере следующими решениями. Вся работа сервера протекает в рамках одного процесса операционной системы. Однако этот процесс имеет свою внутреннюю структуру, элементами которой являются квазипараллельные подпроцессы, называемые lightweight-процессами (в литературе в таком же значении часто используют термин "thread" [5]). Каждому клиенту (Unix-процесс) при подключении к серверу ставится в соответствие один lightweight -процесс, который будет интерпретировать сообщения данного клиента. В ходе интерпретации этот процесс может исполнять операции с дисплеем, реагировать на клавиатуру и мышь, взаимодействовать с другими lightweight-процессами, а следовательно, и другими клиентами. Для построения программ таких процессов NeWS PostScript предлагает удобную и эффективно реализуемую систему понятий.

В следующих разделах дана общая характеристика языка PostScript и его расширений в NeWS. Эта характеристика даст представление о базовых функциональных возможностях NeWS сервера. На практике же сразу после инициализации он предоставляет клиентам существенно более развитый сервис, реализуемый как расширение базовых функций. Эти расширения определяются в целом ряде библиотечных NeWS PostScript программ, выполняемых в процессе инициализации.

Далее будет показано, каким образом должны строиться программы NeWS-клиентов. Сейчас же кратко опишем вспомогательные средства для разработки этих программ на языке Си. Такие средства входят в стандартный состав NeWS.

Язык PostScript

PostScript является примером языка, в котором мощные графические средства встроены в рамки легко расширяемого интерпретируемого языка программирования с простым синтаксисом, универсальной моделью данных и широким спектром возможностей для управления выполнением программ.

Появление языка PostScript (1982 год) было вызвано потребностью в средствах описания процессов автоматического синтеза двумерных тексто-графических изображений на растровых устройствах вывода ин-

формации. В таком качестве он получил достаточно широкое признание. Целый ряд принтеров аппаратно интерпретируют описания документов на языке PostScript [6].

Широкое распространение получили так называемые настольные издательские системы, позволяющие интерактивно конструировать графические документы на экране монитора, изготавливать их твердые копии, передавать их PostScript-описания в другие системы [20,21].

Модель синтеза изображений

Успех PostScript во многом был предопределен свойствами принятой за основу модели синтеза изображений (imaging model [7]). Привлекательность этой модели послужила одной из главных причин использования языка PostScript не только в NeWS, но и в других системах (например, язык Display PostScript на NeXT компьютерах [8,9]).

Представления о формировании изображений, реализованные в языке PostScript, отличаются простотой, полнотой и абстрактностью. В основе лежит stencil/paint-модель, что можно перевести как "раскраска через трафарет". Под трафаретом (stencil) здесь понимается некоторый контур, специфицируемый бесконечно тонкой границей, составленной из гладких кривых в непрерывном координатном пространстве. Раскраска (paint) — это нанесение либо краски одного цвета, либо любого изображения на поверхность отображения. Предполагается, что эта операция всегда выполняется через трафарет. Результирующее изображение соответственно будет иметь контуры, определяемые выбором трафарета.

Такая модель полна, поскольку и линии, и текстовые символы могут быть определены через трафареты. Она проста для понимания вследствие удачного выбора метафоры. Абстрактность этой модели выражается, например, в том, что определение фонта допускает его различные реализации: либо как битовые карты, либо как векторный или гладкий контур. За счет абстрактности модели синтеза изображений PostScript приобрел свойство независимости от характеристик устройств отображения. Эволюция аппаратных средств визуализации не является препятствием для использования и распространения этого языка.

PostScript-программист выражает процесс построения желаемой картины с помощью графических примитивов (операторов) языка.

Трафарет в языке PostScript специфицируется произвольными комбинациями векторов, дуг и кривых высокого порядка с помощью операторов конструирования траекторий (path construction operators). Траектории определяются в терминах точек, специфицируемых координатами на так называемых страницах (page). По умолчанию используется координатная система PostScript, которая может быть изменена операторами линейных трансформаций координатного пространства (coordinates system operators), выполняющими сдвиг начала координат, изменение масштаба, вращение. Построение изображений на странице осуществляется операторами раскраски (painting operators). Построенная страница воспроизводится на устройстве отображения оператором showpage.

Операторы PostScript, как правило, требуют нескольких аргументов, часть из них задается явно, другие же — неявно, как побочный эффект ранее выполненных операторов установки графического состояния (graphics state operators). Среди таких неявных аргументов наиболее существенными являются current page (текущая страница), current path (определяет текущий трафарет), current clipping path (определяет границу рабочей области текущей страницы), current transformation matrix (матрица, задающая отображение координатного пространства пользователя в координатное пространство устройства отображения), current font (текущий фонт, определяющий графические представления текстовых символов). Кроме того, в графическое состояние входят текущее значение цвета, толщина линий, текущее положение в координатном пространстве и другие. В языке предусмотрены операторы для сохранения и восстановления графических состояний.

Учитывая специфику структур данных и механизмов работы с описаниями символов и фонтов, соответствующие операторы выделены в PostScript в отдельную группу (character and font operators). Эти операторы позволяют описывать, выбирать и модифицировать фонты.

Для полноты представления о графических возможностях PostScript перечислим основные операторы из группы painting-операторов: fill (закрашивает область страницы, задаваемую текущей

траекторией), stroke (вырисовывает линию по текущей траектории), show (специальный оператор для вывода изображения символа из текущего фонта), image (помещает на текущую страницу дискретное изображение, полученное либо из внешнего источника, либо синтезированного программно), imagemask (получает изображение аналогично image, но использует его как маску для закраски страницы текущим цветом).

Характеристка синтаксиса и модели вычислений

Наряду с возможностями графических построений PostScript включает в себя как специфичные, так и обычные для языков программирования возможности. Отдельные идеи и элементы PostScript заимствованы из других широко известных языков.

Синтаксис этого языка очень похож на синтаксис языка Forth [10]. Выражения языка записываются в постфиксной нотации, в которой операнды предшествуют операторам. PostScript содержит большое число встроенных операторов, но имена их не зарезервированы и могут быть переопределены программистом. Количество специальных символов очень мало.

К отличительной особенности Post Script относится отсутствие какого-либо синтаксического оформления понятия программы. PostScript-интерпретатор не считывает программу целиком перед ее выполнением, а обрабатывает ее последовательно. Из последовательности символов текстового представления программы (входного потока) по синтаксическим правилам формируются лексемы (tokens), из одной или более лексем создается PostScript-объект (значение данных в памяти интерпретатора) и затем этот объект сразу выполняется. Далее этот процесс повторяется до конца входного потока. При этом, конечно, выполнение программы может иметь побочные эффекты. В частности, они могут выражаться в создании процедурных объектов в памяти интерпретатора, предназначенных для последующих вызовов.

Очевидно, что такой синтаксис и организация выполнения программ очень подходят для использования PostScript в системе NeWS, где программа формируется последовательно во времени как поток сообщений от клиента к серверу.

Модель данных языка включает в себя обычный набор типов данных (числа, строки, массивы, булевы значения, файлы). Одним из основных понятий PostScript является "словарь" (dictionary) — ассоциативная таблица, к элементам которой относятся пары "имя-значение". Наряду с перечисленными типами объектами манипуляций и передач в PostScript-программе могут быть и те элементы, которые обычно рассматриваются как элементы программ (имена, операторы, процедуры). В этом языке любой объект может трактоваться как данные и может быть выполнен как часть программы. За счет этого PostScript приобрел свойства, характерные для языков программирования символьных вычислений, таких, как Lisp.

Каждый PostScript-объект имеет скрытую структуру из трех элементов: тип, атрибуты (литеральный/выполнимый, право доступа) и значение. Эти элементы объекта относительно независимы. Например, два объекта могут иметь один и тот же тип, разделять одно значение, но иметь разные значения атрибутов). Текущее состояние элементов объекта определяет семантику его выполнения.

В основу модели вычислений языка PostScript положены принципы вычислений на стеках. Для представления состояния выполнения программ PostScript-интерпретатор использует четыре различных стека — стек операндов (operand stack), стек словарей (dictionary stack), стек вызовов (execution stack) и стек графических состояний (graphics state stack).

Стек операндов используется для передачи операндов PostScript-операторам и для размещения результатов выполнения операторов. На этот стек могут быть положены любые PostScript-объекты. Контроль типов операндов осуществляется динамически для всех встроенных операторов.

Стек словарей определяет текущий контекст для неявного поиска значений выполнимых имен. Поиск ведется сверху вниз по всем словарям стека до первого определения данного имени. Таким образом, на основе стека словарей реализуется модель динамического связывания имен и значений.

Верхний элемент стека словарей называется текущим словарем (current dictionary). Как правило, именно в этот словарь помещаются определения имен (пары "имя-значение") в ходе выполнения программы.

Однако есть возможности для размещения новых определений в словари непосредственно через их имена. Указанные возможности задания новых определений и переопределения ранее установленных значений, а также операторы изменения стека словарей делают PostScript легко расширяемым языком, а PostScript-программы — очень гибкими. Эти свойства хорошо соответствовали целям разработчиков системы NeWS и, видимо, являлись одной из главных причин выбора PostScript.

На стеке вызовов сохраняются выполнимые объекты (процедуры и файлы). Когда интерпретатор прерывает выполнение одного объекта для выполнения другого, то он кладет прерванный объект на этот стек, а когда завершает выполнение объекта, то снимает верхний объект со стека и продолжает его выполнение.

Как должно быть ясно из сказанного, графическое состояние имеет сложную структуру, состоящую из множества объектов. Текущие значения этих объектов определяют контекст выполнения графических операторов в текущей точке выполнения программы. PostScript-программа может сохранять и восстанавливать графические состояния с помощью двух операторов — gsave и grestore. Для этого и используется стек графических состояний.

Упомянутые стеки полностью независимы, и доступ к ним осуществляется различными средствами. Стек операндов и стек словарей управляются непосредственно выполняемой программой. Наоборот, стеком вызовов полностью управляет интерпретатор, а из программы он может только читаться, но не модифицироваться. Доступ к этим стекам и стеку графических состояний поддерживается соответствующими операторами языка.

В целом возможности PostScript как языка общего назначения определяются семантикой встроенных операторов следующих типов.

1. Операторы манипулирования стеком операндов.
2. Операторы для работы со словарями и стеком словарей.
3. Операторы манипулирования строками и массивами.
4. Арифметические и математические операторы.
5. Операторы сравнения и логические операторы.
6. Операторы изменения и анализа атрибутов объектов, приведения типов объектов.
7. Операторы управления вычислениями.
8. Операторы доступа к файлам.

Расширения языка PostScript в NeWS

В силу иного предназначения язык PostScript оказывается функционально недостаточным для его использования в качестве языка общения клиентов с NeWS-сервером.

Структура NeWS предполагает, что средствами такого языка будут выражаться все аспекты организации и функционирования пользовательских графических интерфейсов распределенных прикладных систем. Соответственно дополнительно к вопросам синтеза и воспроизведения текстово-графических изображений перед разработчиком встают задачи визуального оживления этих изображений, задания требуемой "чувствительности" его элементов к воздействиям пользователя, задания требуемого способа взаимодействия асинхронных клиентов NeWS-сервера.

Только на основе имеющихся встроенных возможностей языка PostScript задачи таких типов не решить. Для этих целей в NeWS PostScript были введены дополнительные понятия, реализуемые через новые типы объектов и соответствующие новые встроенные операторы. К числу основных дополнительных понятий относятся значения следующих терминов этого языка: canvas (холст), process (процесс), event (событие) и interest (интерес). Их содержание раскрыто далее.

Кроме того, на уровень модели данных языка были вынесены объекты, представляющие цвет, графическое состояние и траектории. Для манипулирования этими объектами введены соответствующие типы операторов.

Холсты

Холст замещает понятие страницы стандартного PostScript и трактуется как поверхность, на которой строится изображение обычным набором графических примитивов этого языка. Окна, меню,

pop-up-сообщения, различные элементы управления в графических интерфейсах, реализуемых на базе NeWS, — все основываются на холстах.

Выполнение NeWS PostScript-программы может приводить к созданию множества холстов. Все холсты, параллельно существующие в NeWS-сервере, являются элементами одной иерархической структуры. Ее корень соответствует физическому экрану и имеет имя framebuffer. Этот холст создается оператором createdevice при инициализации NeWS-сервера.

Каждый новый холст создается на некотором уже существующем, который становится его предком. Посредством ссылок на предка и формируется дерево холстов. Дополнительно к связям этого типа в модели холста существуют связи, фиксирующие относительное расположение холстов по оси Z. Их появление вызвано возможностью произвольного наложения холстов друг на друга. Пространство расположения холстов называют пространством двух с половиной измерений (twoand-half dimensions).

Холсту можно придать произвольную плоскую форму и менять ее в ходе выполнения программы; его можно передвигать в пределах холста-предка; для перекрывающихся холстов можно производить упорядочение по оси Z. Каждому холсту сопоставляется своя матрица преобразования координат, которую можно менять соответствующими операторами PostScript.

Построение изображения на холсте можно производить и до того, как будет вызвано его физическое воспроизведение на экране. Воспроизведение и удаление холста на экране могут выполняться многократно. Для того чтобы холст стал виден на экране, должен быть воспроизведен его предок.

Один из атрибутов холста определяет его прозрачность. Если холст прозрачен, то изображение, нарисованное на нем, оказывается на холсте — его предке, а то, что рисуется на самом холсте-предке (или любом другом нижележащем холсте), будет видно через прозрачный холст. Такие холсты используются для создания на нижележащих холстах зон чувствительности к воздействиям пользователя.

В NeWS PostScript для реализации холстов введен новый тип объектов (canvastype) и набор соответствующих операторов. Фактически каждый объект этого типа представляет собой словарь со специальным набором пар "имя-значение", являющихся атрибутами холста. Доступ к этим атрибутам осуществляется при помощи обычных средств PostScript, предназначенных для работы со словарями. Ограничения на доступ указаны в документации по системе NeWS. В графическое состояние интерпретатора языка введена переменная current canvas (текущий холст). Операторы манипулирования холстами выполняются или над текущим холстом, или над холстом, переданным явно через стек операндов.

Далее перечислены основные операторы манипулирования холстами (под холстами здесь понимаются соответствующие объекты языка):

newcanvas — создание нового холста;

createoverlay — создание нового холста специального типа (реализует метафору прозрачной целлофановой пленки);

readcanvas — считывает дискретное изображение в Sun-формате из файла и создает новый холст с этим изображением;

setcanvas — установка нового текущего холста;

currentcanvas — кладет текущий холст на стек операндов;

(eo)reshapecanvas — установка формы холста по текущей траектории;

(eo)clipcanvas — установка границ области отсечения изображений на текущем холсте;

clipcanvaspath — установка текущей траектории по границе текущего холста;

movecanvas — перемещение холста в плоскости XY относительно его предка;

(e)copyarea — копирует часть изображения на текущем холсте, заданную текущей траекторией;

damagepath — установка текущей траектории по границам нарушенных частей изображения на текущем холсте;

getcanvaslocation — возвращает расстояние холста до текущего холста в текущей системе координат;

canvastotop — операторы упорядочения canvastobottom по оси Z

insertcanvasabove перекрывающихся insertcanvasbelow холстов.

В состав операторов раскраски языка PostScript были внесены изменения, относящиеся к работе с дискретными изображениями.

В NeWS операторы image и imagemask не встроенные, а реализованы на базе новых встроенных операторов: buildimage,

imagecanvas и imagemaskcanvas. Аргументы и их интерпретация в операторе buildimage аналогичны оператору image, но результатом будет построение изображения на новом холсте, создаваемом в ходе выполнения оператора. По своему назначению операторы imagecanvas и imagemaskcanvas похожи соответственно на image и imagemask, но получают изображение не из PostScript-процедур, а путем его копирования с заданного холста на текущий холст.

Процессы

Как отмечалось выше, для обработки сообщений каждого клиента NeWS-сервер создает отдельный lightweight-процесс (далее просто процесс). Включением такого понятия в язык NeWS PostScript эта идея получила логичное завершение. Фактически тем самым PostScript был переведен в класс языков параллельного программирования. Однако целесообразность этого следует не только из наличия множества клиентов. В той или иной форме параллелизм потребовалось бы вводить для получения структурированных решений задач взаимодействия с пользователем.

При выполнении программы клиента может быть порождено целое дерево логически параллельных процессов. При этом каждый процесс протекает в своей среде, т.е. имеет свой графический контекст, стек вызовов, стек операндов и стек словарей.

Новый процесс создается вызовом оператора fork с процедурным объектом на стеке операндов. Значение этого процедурного объекта и будет программой нового процесса. Оператор fork формирует исходное состояние среды этого процесса, активирует его и возвращает процессу-родителю объект типа process.

Формирование среды нового процесса производится копированием текущего состояния среды процесса-родителя. При этом среда процесса-родителя частично становится разделяемой с процессом-потомком. Этот эффект возникает вследствие особенностей семантики операции копирования значений составных объектов (массивы, словари) в языке PostScript: копированию подвергается только один уровень вложенности, а значения элементов, являющихся составными объектами, становятся разделяемыми. Таким образом, у процессов появляется возможность взаимодействовать через общую память. Вместе с ней появляются и проблемы исключения одновременного доступа. Для их решения в NeWS PostScript введены средства для блокирования повторного входа процессов в критические секции программ.

К указанным средствам относится новый тип объектов, названных мониторами (monitor objects), и операторы для их создания (createmonitor), проверки текущего состояния (monitorlocked), вызова процедуры через монитор (monitor). Вызов любой процедуры через монитор будет блокировать прохождение через этот монитор других процессов. По завершению процедуры один из задержанных процессов будет разблокирован и сможет войти в свою критическую секцию.

В NeWS реализована non-preemptive-стратегия планирования процессов: однажды начавшись, процесс протекает до тех пор, пока не окажется блокированным. При этом в каждый момент времени активен только один процесс. Блокирование процесса происходит либо неявно (при операциях доступа к файлам), либо в результате выполнения некоторых операторов, перечисленных далее. Отметим, что выбор указанной стратегии планирования lightweight-процессов — это вопрос реализации, который может быть решен иначе в других версиях NeWS.

Подобно холстам, process-объекты реализованы как специального вида словари. Значения, ассоциированные с ключами такого словаря, определяют текущую среду и текущее состояние соответствующего процесса. Доступ к этим значениям осуществляется обычными средствами работы со словарями, но ограничен только чтением.

Для работы с process-объектами используютсяследующие операторы:

breakpoint задерживает текущий процесс;

continueprocess продолжает требуемый задержанный процесс;

currentprocess возвращает process - объект текущего процесса;

fork создает новый процесс-потомок текущего процесса;

killprocess уничтожает процесс;

pause задерживает текущий процесс на время, необходимое для того, чтобы дать шанс выполниться параллельно протекающим процессам;

suspendprocess задерживает требуемый процесс;

waitprocess ожидает завершения требуемого процесса и возвращает верхний элемент с его стека операндов.

При разработке графических интерфейсов возникают ситуации, когда функционирование не только интерфейса в целом, но и отдельных его элементов реализуется не одним процессом, а некоторым поддеревом процессов. Разработка таких элементов и управление ими упрощаются, если имеются встроенные средства манипулирования поддеревьями процессов. Для таких целей в NeWS PostScript введено понятие группы процессов (process group), реализуемое двумя операторами:

newprocessgroup создает новую группу процессов с текущим процессом как ее единственным членом.

Все его процессы-потомки при создании будут автоматически становиться членами этой группы.

killprocessgroup уничтожает требуемый процесс и всю группу процессов, к которой он принадлежит.

Интересы и события

Разделение контекстов и блокирующие операторы являются лишь частью тех средств, которые NeWS PostScript предлагает для реализации требуемых схем взаимодействия клиентов и отдельных PostScript-процессов. Теоретически и они обладают достаточно высокими моделирующими возможностями (можно реализовать, например, такой общий механизм, как семафоры). Однако главную роль среди базовых средств взаимодействия процессов в NeWS играют механизмы, реализующие модель событий.

Целесообразность введения этих механизмов можно объяснить следующими причинами. С одной стороны, в них реализовалось стремление авторов обеспечить модульность и гибкость прикладных систем. Другая причина заключается в том, что в терминах событий удобно представлять активность пользователя во взаимодействии с прикладной системой. Соответственно за счет выбора подходящей модели событий, можно обеспечить единый базис для реализации как требуемых схем взаимодействий внутренних процессов NeWS, так и требуемых способов управления вводом от пользователя.

Характерными особенностями модели событий в NeWS являются введение понятия "интерес" процесса к событиям и возможность передачи через события любой информации. При этом интерес никак не связан с конкретными источниками генерации событий; допускается одновременное сосуществование множества интересов у одного процесса; имеются возможности динамического формирования, выражения и изменения интересов. Если несколько процессов выразили интерес к одинаковым событиям, то каждый из них получит возможность отреагировать на каждое такое событие.

Интересы и события представляются в NeWS PostScript структурированными объектами одного типа (event-объекты), реализованными как словари. Для генерации события надо создать новый event-объект, установить требуемые значения его полей и выполнить оператор передачи события в глобальную очередь механизма распределения событий (sendevent). Аналогично формируется интерес: создается и настраивается event-объект, а затем выполняется оператор выражения интереса текущего процесса (expressinterest). В результате event-объект данного интереса будет помещен в один из списков интересов, которые анализируются механизмом распределения событий. Фактически каждый интерес является образцом событий, на которые процесс будет реагировать. Соответствие события интересу выясняется путем сравнения значений определенных полей их event-объектов по фиксированным правилам. Эти функции возложены на механизм распределения событий.

События отдельных выделенных типов генерируются самим NeWS-сервером в ответ на такие действия пользователя, как перемещение мыши, нажатие или отпускание одной из ее кнопок, нажатие клавиши, переход курсора через границу холста на экране, и другие. При формировании этих событий сервер записывает в их event-объекты информацию о типе события, текущем времени, текущем положении курсора (холст и координаты). Заинтересованные процессы могут использовать эти данные в своих реакциях на события.

В NeWS отсутствует специальный тип таймерных событий, но при формировании любого события можно задать момент времени, когда оно должно произойти. Этой возможности достаточно

для построения различных таймеров. Текущее время и время, когда произошло последнее событие, можно узнать, выполняя операторы currenttime и lasteventtime.

Для ожидания любого события из области своих интересов процесс должен выполнить оператор awaitevent. Этот оператор блокирует процесс до момента поступления соответствующего события, после чего копия event-объекта события будет положена на стек операндов данного процесса, а сам процесс будет разблокирован. Для избежания временных задержек перед выполнением awaitevent можно узнать число свершившихся событий из области интересов текущего процесса с помощью оператора countinputqueue. Анализируя поля полученного event-объекта, процесс выберет способ реагирования на него. Возможен другой подход к реализации реакции на событие — задание процедуры реагирования непосредственно при формировании интереса.

В отдельных случаях может потребоваться изменение или даже уничтожение ранее выраженного интереса. Изменение производится обычным способом, т.е. модификацией соответствующего event-объекта операторами доступа к словарям. Для уничтожения интереса предусмотрен специальный оператор (revoke- interest).

Таково общее представление о способе реализации модели событий в NeWS.

Построение программ клиентов NeWS

Для построения прикладных программ клиентов некоторого общего сервера требуется заранее определенный способ спецификации взаимодействий клиента с сервером, приемлемый в рамках выбранного языка программирования. Этот способ может быть сведен к использованию библиотек, макросредств либо к какой-то другой форме.

Например, построение программ клиентов на языке Си в системе X Window основывается на вызовах процедур библиотеки Xlib [11]. Эту библиотеку называют Си-интерфейсом к X-протоколу связи клиентов и сервера. С точки зрения программиста, Xlib реализует базовые функции оконного сервиса X Window, такие, как подключение к серверу, создание окон, графический вывод, работу с событиями. Фактически же большинство Xlib-подпрограмм транслируют вызовы в запросы уровня X-протокола и передают их по сети серверу. Он и выполняет семантические действия. Таким образом, в системе X Window спецификация взаимодействий с сервером основана на представлении базовых функций X-сервера в терминах Си-процедур и их аргументов. Для программиста изучение этого способа неотделимо от усвоения всей системы понятий о графических оконных интерфейсах, реализованной в X Window.

Аналогично X Window программы клиентов NeWS-сервера можно разрабатывать на различных языках программирования. NeWS также поддерживает определенный протокол связи клиентов с сервером и предлагает свою форму Си-интерфейса к этому протоколу. Однако вследствие использования развитого языка для общения клиентов с сервером функциональная нагрузка и на протокол, и на интерфейс существенно уменьшилась по сравнению с X Window. Разработка NeWS-протокола фактически превратилась в выбор эффективного способа кодирования текстов PostScript-программ. Функции же интерфейса к NeWS-протоколу свелись к поддержке операций создания и разрыва связи с сервером, конструирования и передачи текстов PostScript-программ от клиента к серверу, ожидания и приема ответных сообщений сервера.

NeWS-протокол

Реализация взаимодействий клиентов и NeWS-сервера в ОС Unix основана на sockets-механизме (механизм гнезд) [12]. Такой выбор типичен в ситуациях, когда требуется реализовать взаимодействие процессов в Unix-сетях. Общая схема использования sockets-механизма клиентами NeWS такова.

Для образования канала к серверу клиент должен создать новое гнездо и соединить его с гнездом сервера (адрес гнезда сервера хранится в переменной Unix-среды). После установления соединения клиент может посылать сообщения серверу и принимать сообщения от него. Уничтожая свое гнездо, клиент разрывает связь с сервером. Для выполнения всех перечисленных действий вполне достаточно набора стандартных системных вызовов, соответствующих sockets-механизму.

На уровне NeWS-протокола сообщения, передаваемые от клиента к серверу и в обратном направлении, образуют байтовые потоки. Элементами потока, направляемого серверу, являются либо ASCII-коды текста передаваемой PostScript-программы, либо коды компрессированных лексем этой программы. Эти две формы могут чередоваться в произвольном порядке в одном потоке. Аналогично строится и поток от сервера к клиенту. Однако содержание этого потока полностью определяется той PostScript-программой, которую клиент ранее передал серверу. Элементами содержания могут быть значения любых PostScript-объектов. При этом кодировка каждого значения определяется типом оператора вывода, использованного в программе (ASCII-коды — оператор print, компрессированная форма — tagprint и typedprint).

Введение в NeWS-протокол компрессированной формы передачи PostScript-текстов имеет своей целью только уменьшение временных затрат. В частных случаях достаточно использовать ASCII-представление. Описание правил кодирования лексем занимает менее двух страниц в документации NeWS [1]. В соответствии с этими правилами компрессированными могут быть лексемы, соответствующие целым и вещественным числам, именам встроенных операторов NeWS PostScript, а также некоторым именам, определяемым в программах. Для реализации этих возможностей NeWS-сервер поддерживает одну статическую таблицу для встроенных NeWS PostScript-объектов и ассоциирует с каждым клиентом по одной динамически изменяемой таблице для определяемых объектов. На основе порядковых номеров объектов в этих таблицах и строятся коды соответствующих лексем (один или два байта). Содержание таблицы клиента формируется в процессе выполнения его NeWS PostScript-программы. Для таких целей в NeWS PostScript был введен оператор setfilein-puttoken.

Си-интерфейс NeWS

Взаимодействия клиента с NeWS-сервером можно специфицировать непосредственно в терминах системных вызовов и вызовов функций стандартной библиотеки ввода-вывода ОС Unix. Однако для упрощения программирования на языке-Си в состав программного обеспечения NeWS включены библиотека libcps и препроцессор CPS. Эти средства позволяют программисту оперировать абстракциями более высокого уровня, чем уровень NeWS-протокола.

С помощью библиотеки libcps программист освобождается от обязательного знания деталей организации канала связи клиентов с сервером. Отдельными процедурами этой библиотеки реализуются операции подключения клиента к серверу (ps_open_PostScript) и уничтожения соединения (ps_close_PostScript). Вызвав процедуру ps_flush_PostScript( ), клиент может инициировать принудительную передачу буферизованной части байтового потока к серверу.

Препроцессор CPS определяет удобный способ синтаксического оформления операций конструирования и передачи сообщений от клиента к серверу. В общих чертах этот способ заключается в следующем.

Спецификации требуемых операций взаимодействия клиента с сервером выражаются в гибридной Си-PostScript форме и называются интерфейсными процедурами. Заголовок такой процедуры строится по правилам языка Си, а ее телом является текст PostScript-сообщения. Если список формальных параметров в заголовке не пуст, то их имена могут быть включены в текст сообщения. При этом тип формального параметра должен соответствовать его использованию в PostScript-теле интерфейсной процедуры. В состав допустимых типов входят некоторые Си-типы (int, float, string), а также числа с фиксированной точкой (fixed), лексемы (token) и массивы символов (cstring). Вызов интерфейсной процедуры в Си-программе клиента приведет к подстановке параметров вызова в PostScript-тело этой процедуры и передаче его серверу. Посылаемое сообщение будет компрессированным.

Источниками сообщений, передаваемых клиенту сервером, могут быть любые lightweight-процессы, порожденные данным клиентом. Учитывая асинхронность этих процессов, программирование обработки выходного потока сервера на уровне NeWS-протокола не такая простая задача.

Препроцессор CPS позволяет абстрагироваться от представления выходного потока сервера в виде последовательности байтов или компрессированных лексем. Для этой цели препроцессор CPS определяет специальное синтаксиче-

ское понятие "интерфейсная функция". В виде интерфейсных функций задаются спецификации операций типа "послать сообщение — принять ответ". При этом результаты работы сервера представляются как очередь тэгированных пакетов. Обработка же сообщений сервера основывается на операции проверки соответствия первого пакета очереди требуемому тэгу и операции задержки на время, пока очередь пуста.

Каждый тэгированный пакет имеет структуру последовательности компрессированных лексем, первый элемент которой является тэгом. В качестве тэгов используются целые неотрицательные числа. На стороне сервера тэгированные пакеты формируются lightweight-процессами данного клиента с помощью операторов вывода tagprint и typedprint.

Синтаксически каждая интерфейсная функция состоит из двух частей — интерфейсной процедуры и описания тэгированного пакета. Первая определяет передаваемое сообщение, а вторая — принимаемое. В таких спецификациях параметрами интерфейсной процедуры могут быть и указатели. Из списка имен этих параметров-указателей и значения тэга строится описание пакета.

Вызов интерфейсной функции в Си-программе клиента имеет следующую семантику. На первом этапе выполняются те же действия, что и при вызове интерфейсных процедур. После же посылки сообщения серверу производится анализ очереди пакетов от сервера. Если очередь пуста, то текущий процесс блокируется до поступления любого сообщения от сервера. Если тэг первого пакето очереди не равен тэгу данной интерфейсной функции, то результатом вызова будет логический нуль. Иначе выполняется распаковка пакета и запись его элементов по указателям, заданным в вызове. Возвращаемым значением в этом случае будет логическая единица.

Путем разработки набора интерфейсных процедур и функций программист может создать свой Си-интерфейс у каждого клиента NeWS-сервера. В соответствии с требованиями препроцессора CPS все эти определения должны помещаться в отдельный файл, называемый файлом спецификации интерфейса. В обработке таких файлов и заключается функция препроцессора. В ходе обработки препроцессор CPS транслирует интерфейсные функции и процедуры в форму вызовов подпрограмм библиотеки libcps. В результате создается h-файл, который включается в текст Си-программы клиента директивой include.

Литература

[1] Sample NeWS Manual, Sun Microsystems, Inc., Part No: 800-1973-01, 1987.

[2] Hayes F. and Baran N., "A Guide to GUIs", Byte, Vol.14, No.7, 1989, pp.250-257.

[3] Cole C.T., "Real-Time Unix: Fact or Fantasy?", UNIX REVIEW, Vol.8, No.8, 1989, pp.40-44.

[4] Diehl S. and Wszola S., "Laser Printers Get Personal", Byte, Vol.15, No.7, 1990, pp.138-156.

[5] Warnock J. and Wyatt D., "A Device Independent Graphics Imaging Model for use with Raster Devices", Computer Graphics, Vol.16, No3., 1982.

[6] Software and Peripherals for the NeXT Computer System, NeXT, Inc., 1989.

[7] "The NeXT Computer", Byte, Vol.13, No.12, 1988, pp.250-257.

[8] Баранов С.Н., Ноздрунов Н.Р. Язык Форт и его реализа ции. — Л.: Машиностроение, 1988. -157 с.

[9] Nye A., Xlib Programming Manual, Vol.1, O'Reilly & Associates, Inc., 1988.

[10] Bach M.J., The design of the UNIX operating system, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[11] NeWS Manual, Sun Microsystems, Inc., Part No: 800- 1632-10.

[12] NeWS Technical Overview, Sun Microsystems, Inc., Part No: 800-1498-05.

[13] Pixrect Reference Manual, Sun Microsystems, Inc., Part No: 800-1254-03.

[14] PostScript Language Reference Manual, Addison-Wesley, 1988.

[15] PostScript Language Tutorial and Cookbook, Addison- Wesley, 1985.

[16] Миропольский В., "Язык PostScript и его конкуренты", Компьютер Пресс, Третий выпуск, 1990, сс.18-20.

[17] Hoeber T., "Face to Face with Open Look", Byte, Vol.13, No.13, 1988, pp.286-296.

[18] Smith B., "Personal Iris: The Dream Maker", Byte, Vol.15, No.7, 1990, pp.176-178.

[19] Jones O., Introduction to the X Window System, Prentice-Hall, Englewood Cliffs, NJ.

[20] Goodwin M., "FrameMaker 2.0", PC World, Vol.8, No.5, 1990, pp.160-161.

[21] Diehl S. and Eglowstein, "Is the Typesetter Obsolete?", Byte, Vol.15, No.10, 1990, pp.152-168.

Питер Нортон основал компанию "Питер Нортон Компьютеринг Инк." в 1982 году для распространения первого в мире пакета программного обеспечения для персональных компьютеров - Питер Нортон Ютилитис. Господин Нортон, журналист и компьютерный эксперт, является автором целого ряда книг о персональных компьютерах. Среди них наиболее известны "Внутри IBM PC", "Справочник программиста" и "Внутри OS/2". Кроме того, он является одним из авторов Справочника Питера Нортона по языку для IBM PC и Компаньона Твердого Диска.

Питер Нортон

ВЗГЛЯД В ПРОШЛОЕ, ПЕРСПЕКТИВЫ БУДУЩЕГО

Выступление на открытии выставки "Comtek 91".
Москва, 8 апреля 1991 г.

ВВЕДЕНИЕ

Удивлен сходством той ситуации, в которой вы сегодня находитесь, с тем, что мне пришлось самому испытать в начале семидесятых годов.

В Советском Союзе происходят большие перемены, и вместе с ними у людей возникает все больше возможностей начать свой бизнес. Надежды возлагаются на способных людей с их прекрасными идеями, а также на большое количество потенциальных покупателей с их неудовлетворенными потребностями.

И успех возможен как для малых предприятий с довольно скромными ресурсами, так и для большого бизнеса.

Советские предприниматели имеют сейчас те же заманчивые перспективы, что и в свое время, или мои коллеги — Митч Кэйпор, который написал Lotus 1-2-3, или Стив Джобс, основавший Apple Computers, или Билл Гейтс, президент Microsoft — и сотни других людей, ставших первопроходцами американской индустрии микорокомпьютеров.

Когда я начал заниматься программированием в начале семидесятых, персональных компьютеров было очень мало. Появление IBM PC в 1982 году способствовало быстрому росту отрасли. Большая часть инфраструктуры, которую мы сейчас имеем в США с ее доступом к капиталу, каналами распространения, образованием и так далее, начала формироваться именно в то время.

Мы знали, что нам нужно сделать, чтобы решить те сложные задачи, которые мы ставили перед собой, но вряд ли кто-нибудь представлял себе, к каким грандиозным результатам приведет развитие новой индустрии.

У вас же есть то, чего не было у нас, то, что может быть очень ценным на этом этапе развития советской экономики. У вас накоплен колоссальный опыт в импровизации, в умении работать только с тем, что у вас есть под рукой, в способности полагаться на самих себя. У вас есть то, что называется "жи-

тейской мудростью". Знать, как извлечь выгоду из того, что есть, и выйти из любой ситуации — это отличный опыт, который можно с успехом использовать, чтобы начать свое дело.

И поскольку история повторяется, я бы хотел поделиться с вами своим опытом.

НАЧАЛО

Когда вы начинаете свое дело, вам надо быть готовым к тому, чтобы засучить повыше рукава и делать все самостоятельно. Когда я написал свою первую программу "Unerase", практически никакой помощи со стороны не было, чтобы попытаться получить какую-то выгоду от существования этой программы. У меня не было ни опыта, ни образования в области бизнеса. Все, что у меня было, — мои сбережения. Но я чувствовал, что сделал отличную программу, и мне не хотелось останавливаться. И я хотел сделать все сам.

Заказы на первые программы в 1982 году я выполнял у себя дома, используя персональный компьютер для копирования программ на гибкие диски, а затем шел на почту и отправлял их своим заказчикам.

СОЗДАНИЕ ПРОГРАММЫ: ПЕРВЫЙ РАУНД

Создание хорошей программы — трудное дело. Существует два подхода, которые вы можете при этом использовать. Оба могут быть успешны. Все зависит от обстоятельств.

Подход первый — это полагаться во всем на себя. Второй — полагаться на вашего заказчика.

Если вы верите в себя, если интуиция подсказывает вам, что вы знаете, что может понадобиться людям, — следуйте вашей интуиции. Если интуиция вас не обманывает, тогда, положившись на себя, вы непременно придете к успеху.

Второй подход — положиться на заказчика. Спросите ваших потенциальных пользователей, что им нужно. Вы узнаете много интересного, поговорив с каждым из них.

Преимущество первого подхода заключается в том, что у вас появляется хорошая интуиция, некое видение будущего — вы можете знать, что пользователям понадобится завтра, в то время как сами пользователи могут этого не знать. Очень часто пользователей беспокоят проблемы вчерашнего дня, а главная задача для нас, бизнесменов — создавать программы, которые будут нужны завтра.

Например, в 1982 году, когда я написал свою первую программу "Unerase", позволяющую восстановить случайно стертый файл, никому такие программы, казалось, не были нужны. Но я-то знал, какую ценность для меня и моих друзей представляет написанная мною программа. Спустя некоторое время, когда все поняли, насколько важно иметь возможность восстановить утерянные данные, "Unersase" фактически сформировала новый сектор рынка программных средств для PC, который называется рынком сервисных программ.

Преимущество второго подхода в том, что вы узнаете, что представляет собой ваш потенциальный заказчик, каковы его потребности. Вы узнаете мнения многих людей. Такой подход мы называем "управляемый пользователем".

Оба эти подхода могут работать. Вам надо понять, какой из них больше подходит к определенной ситуации.

СОЗДАНИЕ ПРОГРАММЫ: ВТОРОЙ РАУНД

Совершенно очевидно, что создание одной программы не может обеспечить длительный успех. Параллельно с разработкой первого продукта необходимо думать о том, что будет дальше. Существуют два способа двигаться дальше. Первый — улучшить существующий продукт, второй — создать новый.

Первая опция: усовершенствовать продукт. Создание новой версии существующего продукта — если он пользуется успехом — одна из наиболее важных конкурентоспособных характеристик, которую вы можете использовать в условиях свободного предпринимательства. Во-первых, пользователь уже имеет ваш продукт, и теперь потребуется прилагать так же много усилий для продажи новой версии. Пользователь вложил время и деньги в изучение программы и знаком с ней. Во-вторых, вы используете созданный продукт и добавляете к нему новые

функции. Это гораздо быстрее и дешевле, чем начинать продукт с нуля.

Мы не раз обновляли и усовершенствовали наши сервисные программы. Каждая новая версия предстает более мощной, качественной, функционально насыщенной. Фактически только в прошлом году мы выпустили Nortpn Utilities 5.0. Это пятая большая версия той самой первой "Unerase", которую я создал в 1982 году, а что касается Norton Commander — мы сделали уже третью версию.

Так что, как вы сами видите, усовершенствование продукта, который хорошо зарекомендовал себя, может быть весьма прибыльным делом.

Вторая опция: создать новый продукт. Вы можете создать продукт, который, несмотря на конкуренцию, не обновляется производителем. И лучшее, что вы можете сделать в этой ситуации, — предложить лучшую версию конкурентного продукта, а не создавать полностью новый.

Мы, специалисты по высоким технологиям, зачастую сосредотачиваемся только на самых современных разработках, охотимся за наиболее совершенными продуктами и восхищаемся новыми технологиями. Но, как я узнал, вашего пользователя просто так не вдохновишь.

Бизнесмену стоит помнить, что большинство пользователей персональных компьютеров выполняют общие задачи и используют базовые продукты типа Lotus 1-2-3. У них нет ни времени, ни желания, чтобы перейти на использование современной технологии. Помните об этом при создании нового продукта.

ИНТЕЛЛЕКТУАЛЬНАЯ СОБСТВЕННОСТЬ

Это еще одна тема, о которой в последнее время говорят все больше и больше. В США этот вопрос также весьма активно обсуждается — особенно применительно к программному обеспечению. Я знаю, что многие из вас имеют копии некоторых моих программ, и очень приятно видеть, что они так популярны здесь.

Когда я начинаю распространять продукт, я хочу, чтобы мои пользователи могли извлечь из него максимальную пользу. Но это означает, что им нужно руководство по эксплуатации, сопровождающее программу. Им надо попробовать поработать со всеми функциями. Если им понадобится дополнительная помощь, они обратятся в отдел поддержки пользователей, чтобы получить дополнительную подготовку.

МАРКЕТИНГ И ПРОДАЖА: БОЛЬШЕ, ЧЕМ ПОЛОВИНА ЗАДАЧИ

Создание хороших программ — это только часть той сложной задачи, с которой мне пришлось столкнуться после того, как я написал "Unerase". Одна из любопытных особенностей свободной экономики в том, что маркетинг и продажа продукта — это более чем подзадачи. У нас даже поговорка есть, что даже самый прекрасный программный продукт не сможет продать сам себя в условиях свободного предпринимательства.

Когда я предложил свою первую программу, я прилагал массу усилий, чтобы довести ее до пользователей. К тому времени я уже несколько лет работал с компьютерами, но только тогда начал использовать IBM PC, распространяемый на коммерческой основе. Как только я научился работать с компьютером, я решил поделиться своими навыками со всеми пользователями посредством распространения информационного листка. Я назвал этот листок "шпаргалкой".

Шпаргалки содержали много практической информации и имели, конечно же, рекламу моих собственных продуктов на обратной стороне каждого листка. Каждые два месяца я посылал мои шпаргалки по почте в компьютерные клубы и магазины — например, Computerland. Главной целью было сделать мои программы известными и внушить пользователям доверие к ним.

Важность маркетинга поистине трудно переоценить. Я использовал еще одно средство для популяризации моих продуктов. Я понял, что технологическим продуктам недостает человеческого тепла. Нужно было, чтобы мой продукт имел эти человеческие черты, поэтому я поместил на него свое имя, назвал своим именем компанию, а на коробках начали печатать мои фотографии.

Пользователь как бы получает сообщение: "За этой программой стоит

не безликая корпорация, а реальный человек, который знает, что делает".

Это всего лишь два примера из тех средств маркетинга, которые я использовал. Эти средства также зависят от конкретной ситуации и, конечно, от ваших собственных возможностей. Оказалось, что я могу хорошо объяснить пользователю-новичку технические аспекты программы — вероятно, потому, что я хорошо помню свой собственный опыт работы на IBM PC. И писать те самые шпаргалки было для меня вполне естественным занятием.

УПРАВЛЕНИЕ БИЗНЕСОМ: ПРОСТО ТЕХНИЧЕСКИХ ЗНАНИЙ НЕДОСТАТОЧНО

Я говорил немного о важности маркетинга и продаж, что подвело меня к другой теме: важности разумного управления бизнесом.

Когда я основал Peter Norton Computing в 1982 году, у меня не было никакого коммерческого образования. Я занимался программированием для аэрокосмической промышленности в Южной Калифорнии. Имел математическое образование, накопил приличные знания в информатике. Но у меня не было ни малейшего представления о том, как начать свой бизнес и какие надо прилагать усилия, чтобы он был успешным.

Мне приходилось одновременно изучать теорию и практику бизнеса, надо было знать все: от бухгалтерии и управления персоналом до маркетинга и продаж.

Свое образование в области управления я получал "на работе", следуя методу проб и ошибок, чтобы понять, что работает и что нет. Я читал все, что попадало мне в руки, — Fortune, Forbes, Business week, и вообще все, что тогда было написано о бизнесе. И говорил с кем только мог.

Мораль: учитесь управлять — учитесь, где можете и как можете.

ОЦЕНКА СВОЕЙ СИЛЫ

Небольшие компании обычно основываются умными, талантливыми и опытными людьми. Я бы хотел посоветовать вам реально оценить свои силы и найти людей, которые располагали бы опытом в других областях.

И когда вы найдете этих талантливых людей, предоставьте им свободу действий.

В некоторых случаях, развивая это направление, можно соединить опыт двух компаний. В середине прошлого года я объединил свою компанию с Symantes Corporation. Мы объединили опыт создания сервисных программ и хорошо действующую торговую сеть. Symantes выпускает и другие программные продукты — управление проектами, коммуникации, языки программирования.Оценка возможностей компании не менее важна, чем оценка возможностей каждого отдельного человека.

РАЗВИТИЕ ТЕХНОЛОГИЙ

Шесть лет назад я говорил в Болгарии и повторю сейчас: использование персональных компьютеров сулит большую выгоду малому и среднему предпринимательству, но предпринимателям не надо обязательно иметь 386 или 486 микропроцессор, которые так нужны нам — специалистам в области технологий.

Бизнесмен может использовать доступную, простую в эксплуатации и сравнительно дешевую систему. Вы можете значительно улучшить свою производительность при минимальных затратах, приобретя компьютер на 286 микропроцессоре.

Я высоко ценю ваше умение в импровизации, умение использовать жизненный опыт. Это очень важные качества для того, чтобы начать бизнес, чтобы стать преуспевающим предпринимателем.

В заключение, подводя некоторый итог моему опыту, я хотел бы сказать, что если вы действительно заинтересованы, готовы идти на риск, если вы изобретательны — вы добьетесь успеха.

♥♥♥

ВНИМАНИЮ АДМИНИСТРАЦИИ ВУЗОВ, ТЕХНИКУМОВ, ПТУ, СПЕЦШКОЛ:

НОВЫЕ ОБУЧАЮЩИЕ ПРОГРАММЫ КОМПЬЮТЕРНОЙ ФИРМЫ "МИГ"

Позволят вам сэкономить 25000 - 30000 рублей

на лабораторном оборудовании для курсов "ЭЛЕМЕНТНАЯ БАЗА ЭВМ", "УЗЛЫ ЭВМ", "МИКРОПРОЦЕССОРЫ". Осциллографы, источники сигнала и микропроцессорные стенды вам не понадобятся - их достоверно заменят моделирующие программы для локальных сетей КУВТ-86 (БК-0010Ш), КУВТ-86М/УКНЦ-01/ (БК-0011М), УКНЦ.
Быстрота освоения и простота использования, встроенные подсказки и приличный дизайн - вот отличительные черты нового программного продукта. Мощный, продуманный сервис преподавателя позволяет проводить лабораторные работы, не поднимаясь со стула!

ПРОЕКТ "СОКРАТ" КОМПЬЮТЕРНОЙ ФИРМЫ "МИГ" - ЭТО ФАНТАСТИКА!

Проектирование модулей ЭВМ на базе библиотеки логических элементов, компоновка и трассировка печатных плат, варьирование входными сигналами, расчет временных диаграмм созданного на экране компьютера электронного модуля - это "СОКРАТ".
Изучение работы 16-разрядного микропроцессора, на основе которого построены ваши ЭВМ, принципов обмена МП с внешними устройствами на примере ОЗУ, показ сигналов системной магистрали, освоение системы команд МП К180ИВМ1 и, самое важное, получение навыков программирования на АССЕМБЛЕРЕ PDP11 за 6 - 8 часов - это "СОКРАТ".
Преподаватели перечисленных предметов! Эффективность обучения путем погружения в изучаемую среду исключительно высока. Сетевой сервис и прилагаемые варианты заданий избавят вас от рутины, освободив время для творческого труда.

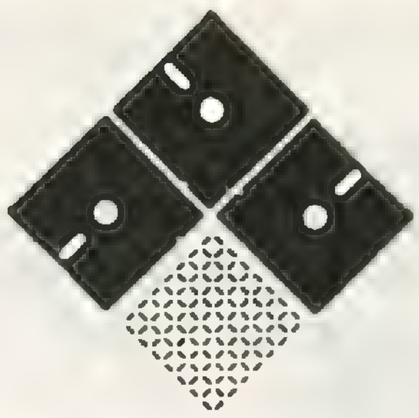
БЕСЕДЫ С "СОКРАТОМ - ЛУЧШЕЕ ВРЕМЯПРОВОЖДЕНИЕ ДЛЯ ВАШИХ ПОДОПЕЧНЫХ!

*Цена пакета - 1240 руб. * При оптовых покупках прогрессирующая скидка * Защита от копирования * Поставка после оплаты*

Демонстрационная версия системы высылается наложенным платежом 120 руб. За дополнительную плату возможен перевод диалога с пользователем на государственные языки суверенных республик.
Убедить в своей неплатежеспособности и добиться льготной продажи, а также получить консультации можно по телефону (095) 327-94-59
Гарантийное письмо с указанием типа ЭВМ, формата записи с приложенной копией платежного документа направлять по адресу:

103031, Москва, а/я 815, р/сч. 609139 в Перовском отд. ЖСБ МФО 201735, код банка Н5.

♠ ♠ ♠



Система Турбо 3Си объединяет в себе мощный компилятор языка Си и интегрированную среду для разработки и отладки программ. Необходимость такого рассказа видится в следующем. Скорее всего, вы заняты изучением языка программирования Си или планируете в ближайшем будущем взяться за это ответственное дело. При изучении языка, тренировке в написании и отладке на ЭВМ собственных программ возникнут по меньшей мере два вопроса: как пользоваться имеющимся компилятором и насколько «понимаемая» данным компилятором версия языка Си отличается от других версий. Эти важные вопросы рассмотрены в статье.

А.А.Ковалев

ТУРБО СИ ДЛЯ НАЧИНАЮЩИХ

В выпуске "Язык программирования Си" (см. "ВТ и ее применение" N 4, 1991) мы начали публикацию статьи "Турбо Си для начинающих". Здесь приводим ее окончание.

Модификатор volatile используется для целей, практически обратных. Он говорит о том, что значение переменной может быть изменено способами, которые программа не специфицирует явно. Например, это может быть внешнее воздействие вроде программы прерывания или порта ввода/вывода. Специальное объявление переменной такого характера предупреждает компилятор, что не следует делать никаких предположений относительно значения рассматриваемой переменной при оценке условных выражений и т.д., ибо ее значение в любой момент может измениться.

Другие важные расширения Турбо Си

Существует несколько дополнительных расширений языка Си, которые делают программу более читаемой. Эти расширения заменяют «анахронизмы» ранних версий Си и позволяют писать более эффективные, красивые и мобильные программы. Опишем некоторые из них.

Строковые литералы. В классическом Си для получения длинной строковой переменной в программе используются символы продолжения или некоторые разновидности конкатенации. В Турбо Си можно просто разбить большую строковую переменную на несколько строк следующим образом:

```
main()
{
    char *msg;
    msg="Раньше все это потребовало"
        "бы от Вас, некоторых ухищрений"
        "для того чтобы записать длинную"
        "строку в одну переменную."
        "Сейчас даже не нужно шевелить"
        "мозгами для того, чтобы запихать"
        "в переменную целое"
        "стихотворение."
    printf ("%s",msg);
}
```

Шестнадцатиричная символьная константа. В классическом Си ключевые последовательности, определяющие конкретные символы в коде ASCII, представлены в восьмиричной системе счисления. Это обусловлено тем, что в начале развития машин двоичные числа обычно представлялись в восьмеричной форме. В настоящее время большинство компьютеров используют шестнадцатиричную форму для представления двоичных чисел. Поэтому современный Си позволяет объявлять символьные константы в шестнадцатиричной системе счисления. Основная форма шестнадцатиричной константы имеет вид: '/xDD', где DD представляет одну или две шестнадцатиричные цифры (из набора 0..9,A..F).

Эта ключевая последовательность может быть прямо присвоена перемен-

ной типа char или может быть вложена в строку: ch = '/x20'.

Типы со знаком. Классический Си подразумевает, что все int-типы являются типами со знаком. Для задания обратного используется модификатор типа unsigned. По умолчанию переменные типа char считаются signed (со знаком), что подразумевает изменение их значений от -128 до 127. Однако в настоящее время большинство компиляторов Си, реализованных для современных моделей компьютеров, воспринимают тип данных char как unsigned. В связи с этим для повышения мобильности программ на Турбо Си в компилятор пакета включена опция, позволяющая компилировать тип данных char как unsigned по умолчанию. В противном случае можно объявить символьную знаковую переменную как signed char.

Правила видимости в Турбо Си

Турбо Си допускает более «вольное» обращение с неуникальными идентификаторами, чем стандарт K&R. Различаются четыре класса идентификаторов:

1. Имена, описанные с помощью слова typedef, переменные, члены перечисления (в переменных типа enum) должны быть уникальными внутри того блока, в котором они описаны. Идентификаторы, которые объявлены как внешние, должны быть уникальны среди соответствующих идентификаторов.
2. Структура и объединение должны быть уникальны внутри блока, где они описаны.
3. Имена переменных, входящих в структуру (членов объединения), должны быть уникальны в структуре (объединении), в которой они описаны. Не существует никаких ограничений на тип или смещение для членов с одинаковыми номерами в различных структурах (объединениях).
4. Операторы goto на заданную метку должны использоваться только в пределах той функции, где объявлена данная метка. Имя метки должно быть уникальным для функции, в которой она объявлена.

Таковы правила видимости, приводимые в Руководстве по Турбо Си. Старайтесь следовать данным рекомендациям во избежание путаницы и ошибок в программах.

Модели памяти Турбо Си. Ключевые слова near, far и huge

Турбо-Си позволяет использовать различные модели памяти. Всего их шесть, как и определено стандартом ANSI.

Напомним, что модель памяти, которую вы используете, определяет внешний формат указателей на данные. Если используется «малая» модель данных (tiny, small, medium), все указатели имеют длину только 16 бит и задают смещение относительно регистра сегмента данных (DS). Если используется «большая» модель памяти (compact, large, huge), все указатели на данные имеют длину 32 бита и задают как адрес сегмента, так и смещение.

В некоторых случаях, когда используется один размер модели данных, можно объявить указатель с размером или форматом, отличным от используемого по умолчанию. Это делается с помощью модификаторов near, far и huge.

Библиотеки Турбо Си

Понятие библиотеки и библиотечных функций, используемых Турбо Си, практически соответствуют классическому K&R. Турбо Си имеет ряд библиотек, в которых содержатся функции для выполнения наиболее часто встречающихся задач. Разработчики Турбо Си включили в систему библиотеку, превышающую по составу определенную в стандарте ANSI. Эта библиотека содержит функции самого разнообразного назначения.

В начале знакомства с Турбо Си можно ограничиться теми функциями, которые известны из трудов K&R. Однако в последствии потребуется детальное знакомство с возможностями библиотек. Перечисление и краткая характеристика имеющихся в них функций представляют собой объемный фолиант. Такая информация представлена в руководстве по Си, она же доступна при работе в интегрированной среде Турбо Си в виде системы подсказки (см. далее).

В комплект поставки входит несколько версий библиотек Турбо Си (по

одной на каждую модель памяти). Кроме того, существуют отдельные математические библиотеки для эмуляции процессора 8087. Условием нормальной работы Турбо Си является «знание» компилятором мест расположения библиотек нужной модели памяти. Эти «места» (каталоги) определяются с помощью опции Options основного меню. Как это сделать, мы расскажем при рассмотрении запуска вашей первой программы.

Projects — Проекты

Турбо Си предоставляет обширные возможности по созданию достаточно сложных программ. Одна из них — наличие опций поддержки так называемых проектов — Си-программ, состоящих из нескольких файлов. О том, как создавать проекты и что с ними делать, мы расскажем отдельно. Сейчас просто введем это понятие для того, что, бы при изучении опций меню Турбо Си вас не смущало незнакомое слово.

А теперь перейдем к конкретным делам.

ПРАКТИЧЕСКИЙ ШАГ: УСТАНОВКА ТУРБО СИ

Может случиться так, что Си уже установлен в машину — это несколько ранее сделали ваши коллеги. В таком случае прочитайте этот раздел «про запас» — может, информация окажется полезной.

Вообще говоря, Турбо Си можно установить на компьютер двояко: «быстро и небрежно» или «согласно инструкции». Рассмотрим оба случая, хотя второй выглядит явно предпочтительнее.

Комплект поставки Турбо Си и установка «по инструкции»

Фирма "Борланд" и ее дилеры предлагают Турбо Си в виде набора так называемых дистрибутивных дискет и соответствующей документации (все это заключается в большую коробку с красивым оформлением). В этой коробке можено найти руководство пользователя, в начале которого на хорошем английском языке подробно рассказывается о необходимых действиях по установке и изучению Турбо Си.

Из дистрибутивных дисков следует выбрать диск, помеченный как INSTALL/HELP (Инсталляция/Помощь). На этом диске обнаруживаем файл с именем INSTALL.EXE. Это так называемый инсталлятор — специальная программа для создания на диске компьютера необходимых каталогов, копирования в них соответствующих файлов пакета Турбо Си и установки нужных параметров в самой системе.

Запустите эту программу. Появляется экран инсталлятора Турбо Си. Ознакомившись с приветствием и авторскими правами фирмы, нажимайте клавишу ENTER (здесь и далее для прекращения процесса установки следует нажать ESC). После появления следующего приглашения вставьте (если ранее она была изъята) дискету INSTALL/HELP в дисковод A, и снова нажмите ENTER.

Следующий шаг более важен — нужно определиться: что, собственно, следует делать. Программа выведет на экран меню из трех элементов:

Install Turbo C on a Hard Disk — установка Турбо Си на жестком диске;

Update Hard Drive Copy of Turbo C 1.5 to Turbo C 2.0 — для дополнения старой версии Турбо Си;

Install Turbo C to a Floppy Drive — установить Турбо Си на гибкие диски.

Мы рассмотрим случай начальной установки Турбо Си на компьютер с жестким (винчестерским) диском. Выберите первую опцию меню и нажмите ENTER.

Вслед появляется новое меню с именами тех каталогов, в которые будут копироваться файлы Турбо Си. Согласитесь с предложенными именами. Кроме того, инсталлятор запрашивает «модель памяти» (Memory Modes). Согласитесь с параметром, предлагаемым «по умолчанию» (модель small).

Итак, выбирайте опцию Start Instalation и нажимайте на ENTER. Теперь нужно следить за приглашениями системы: в середине экрана появляется окошко, где указывается имя требуемого в данный момент диска (оно пишется большими буквами после фразы Please Insert the ...). Вста-

вив соответствующий диск, нажимайте ENTER и ждите следующей информации. Инсталлятор должен выполнить всю работу без замечаний. Но что делать, если вы перепутали и вставили не тот диск, который требуется? Ничего страшного, просто появится красное окошко с соответствующим предупреждением. Поменяйте диски и нажмите R — retry (повтор).

Установка копированием

В начале создайте каталог Турбо Си. Назовите его как угодно, например /TC. Второй шаг заключается в копировании всего содержимого имеющихся дискет в этот каталог.

Затем в каталоге /TC создайте два подкаталога с именами LIB и INCLUDE. В подкаталог LIB перепишите все библиотеки и загрузочные модули (т.е. все файлы .LIB и C0x.OBJ). В подкаталог INCLUDE перемещаются все включаемые файлы (с расширением .H). Кроме того, стоит в отдельные каталоги переписать файлы с документацией и примерами (расширения *.DOC и *.C соответственно).

После выполнения всех копирований и перемещений убедитесь в следующем: в каталоге /TC должны содержаться как минимум следующие «главные файлы» системы Турбо Си:

TC.EXE	Интегрированная среда Турбо Си;
TCHELP.*	Файлы системы подсказки Турбо Си;
TCCONFIG.NC	Файл конфигурации для интегрированной версии;
TCC.EXE	Версия Турбо Си, работающая с командной строкой;
TCCONFIG.TC	Конфигурация для транслятора с командной строкой.

После установки система должна «знать» имена некоторых каталогов. Они указываются в опции OPTIONS/DIRECTORIES интегрированной среды. Несколько позднее мы расскажем, как это сделать.

Выделение каталога для работы

Последний важный момент в установке пакета Турбо Си. Стоит определиться, в каком каталоге будут находиться ваши собственные Си-программы.

Мы предлагаем следующий способ: создавайте отдельные и собственные рабочие каталоги для своих программ. Сделать это можно по-разному. Если вы работаете в Турбо Си в одиночестве, рабочий каталог можно назвать /WORK и разместить его где угодно — можно в каталоге /TC.

Чтобы все файлы скапливались именно в каталоге WORK, он должен быть текущим для системы Турбо Си. В принципе можно запустить Си из каталога /TC, а затем с помощью опций интегрированной среды FILE/CHANGE DIR (Файл/Смена каталога, см. далее) установить новый текущий каталог. Но гораздо удобнее сразу запускаться из своего рабочего каталога.

Для этого файл TC.EXE должен быть «видимым» для MS-DOS. Достичь это можно указанием «пути» в файле AUTOEXEC.BAT — в строку с командой PATH добавьте путь C:/TC.

Можно сделать еще более элегантно. Дополнительные пути в MS-DOS снижают быстродействие компьютера. Поэтому стоит сделать специальный командный файл (batch-filec) для выполнения всех работ по запуску Турбо Си. Предлагаем назвать этот файл TC.BAT и поместить его либо в корневой каталог C:/, либо в первый каталог, указанный в команде PATH файла AUTOEXEC.BAT. Наш командный файл должен содержать следующие команды:

```
@ECHO OFF
2.0!
C:
CD C:/TC/WORK
C:/TC/TC.EXE
```

Первые две команды связаны с выводом на экран сообщений. Затем устанавливается текущий диск C: и ваш рабочий каталог TC/WORK, после чего дается команда на запуск самого Турбо Си. Все очень просто и эффективно! Естественно, нужно определиться с именами — имена каталогов должны соответствовать вашей системе.

Что делать, если Турбо Си используют несколько человек? Для каждого

программиста следует выделить свой рабочий каталог. Очевидно, здесь следующее решение: создайте каталог /WORK, как рассказывалось выше, а в нем — несколько подкаталогов с именами, ассоциирующимися с именами пользователей. Пусть это будут /WORK/VANIA и /WORK/MANIA. Соответственно должны быть написаны и различные командные файлы с соответствующими именами:

Wania.bat

```
@ECHO OFF
ECHO Турбо Си для Вани!
C:
CD C:\TC\WORK\VANIA
C:\TC\TC.EXE
```

Mania.bat

```
@ECHO OFF
ECHO Турбо Си для Машеньки!
C:
CD C:\TC\WORK\MANIA
C:\TC\TC.EXE
```

И еще важный момент. В корневом каталоге C:/ должен быть файл конфигурации системы — CONFIG.SYS. В этот файл помещена строка FILES=20 (или больше 20). Если ничего подобного нет, напишите, если нет самого файла — создайте (что должно быть записано в файле конфигурации ОС — см. литературу по MS-DOS).

Вроде бы все! Турбо Си установлен, и все готово к запуску. Если вы что-либо меняли в файлах AUTOEXEC.BAT и/илиCONFIG.SYS, перезагрузите компьютер. А теперь подготовимся к самому интересному — запуску Турбо Си.

СРЕДА ТУРБО СИ: ПЕРВЫЕ ШАГИ

Два режима работы Турбо Си

Турбо Си может функционировать в двух различных режимах. Первый режим — это так называемая интегрированная операционная среда Турбо Си, одно из интереснейших «творений» фирмы "Борланд". Работая в интегрированной среде, с помощью нажатия функциональных клавиш и выбора опций из меню можно редактировать, компилировать и выполнять разрабатываемые программы. Данный режим обеспечивается файлом TC.EXE. В другом режиме (так называемый режим командной строки) используется более традиционный подход, когда сначала используется редактор, затем компилируется программа, организуется ее связь с другими программами и, наконец, запускается на выполнение. Этот режим соответсвует методике подготовки Си-программ, которую вы, возможно, использовали на машинах типа СМ, ДВК и др. Этот режим обеспечивается файлом TCC.EXE.

Мы будем рассматривать режим интегрированной среды по целому ряду причин. Во-первых, это позволит познакомиться с современной технологией разработки математического обеспечения. Во-вторых, в таком режиме просто легче работать.

Интегрированная среда: идеология построения и методы работы

Попробуйте запустить Турбо Си. Если вы создали для этого специальный командный файл — наберите на клавиатуре его имя и нажмите на ENTER. Если же нет — наберите TC, имя, и также нажмите клавишу ENTER. На экране появляется картинка, которая называется либо экраном основного меню, либо экраном интегрированной среды — как вам будет угодно.

Экран состоит из четырех частей (сверху вниз): **Основного меню; Окна редактора; Окна сообщений компилятора; Строки функциональных клавиш.**

Посреди экрана вначале присутствует окошко серого цвета с информацией о версии Турбо Си и правах фирмы "Борланд". Нажмите любую клавишу — окошко исчезнет, и основной экран приобретает вид, полностью соответствующий рис.2. Теперь самое время немного разобраться, что к чему.

В **окне редактора**, очевидно, должен находиться редактируемый текст. Как правило, это текст Си-программы, хотя никто не запрещает написать с помощью редактора какое-либо письмо или служебную записку.

В верхней части экрана находится **основное меню**. Оно используется либо для указаний каких-либо действий над разрабатываемой программой (например, редактирование, компиляция, выполнение, отладка и т.д.), либо для установки параметров системы Турбо Си.

Некоторые из опций основного меню вызываются при работе в редакторе с помощью так называемых горячих ключей — специально оговоренных клавиш. Именно эти клавиши перечислены в самой нижней строке экрана — строке быстрой подсказки (функциональных клавиш). Обратите внимание: в строке указывается набор только тех клавиш, которые уместны для данного состояния Турбо-системы. При изменении состояния строка также меняется.

Окно сообщений компилятора содержит информацию о произведенных с программой действиях и выявленных при этом ошибках. В момент загрузки и при последующем редактировании текстовых файлов можно ожидать, что окно сообщений будет оставаться пустым.

КОНТЕКСТНО-ОРИЕНТИРОВАННАЯ ПОДСКАЗКА

Отличительной особенностью программных продуктов фирмы "Борланд" является наличие подробной и хорошо написанной документации, а также встроенной системы помощи (подсказки) — HELP.

Вызов системы помощи осуществляется нажатием клавиши **F1**. Система помощи автоматически настраивается на контекст. Это означает, что выводимая на экран информация будет связана с выполняемой вами работой. Для выхода из системы помощи нажмите клавишу **ESC**.

Можно получить подсказку относительно значения того или иного ключевого слова. Для этого, работая с текстовым редактором, установите курсор на интересующее вас слово и нажмите клавиши **CTRL-F1**.

Вызвав подсказку, можно просмотреть ее с помощью клавиш PgUp и PgDn (если информация по данному вопросу не умещается на экране). Можно вызвать оглавление системы подсказки, нажав еще раз F1. А можно восстановить тот экран, который был прочитан ранее (всего до 20 экранов) — клавиши ALT-F1.

Последнее интересное свойство — это использование элементов так называемого гипертекста. Обратите внимание на то, что в ряде разделов подсказки некоторые слова выделены другим цветом, а одно даже отсвечено курсором. С помощью клавиш управления курсором последний можно перемещать между выделенными словами. Для чего это сделано? Все очень просто — выделенное слово содержит ссылку на соответствующий раздел подсказки, выбрав это слово курсором и нажав на ENTER, переходим в соответствующий раздел.

ОРИЕНТИРОВАНИЕ В МЕНЮ ТУРБО СИ

Выбор элементов меню

Во-первых, нужно запомнить — при работе в редакторе меню активизируется нажатием **F10**. Признак активности меню — наличие в нем темного курсора (отметки), который можно передвигать влево-вправо для выбора соответствующей опции. Для выбора опций можно нажать начальную заглавную букву из названия опции. Нажатие Esc приводит к выходу из меню в режим редактирования текста или возврату к другому ранее активному окну.

Кроме того, при работе с текстовым редактором можно сразу выбрать нужную опцию меню нажатием клавиши ALT и первой буквы названия опции (ALT-F для File, например).

Меню имеет многоуровневую структуру — выбор некоторых опций с последующим нажатием на ENTER приведет к появлению «выпадающего» подменю, идея работы с которым все также проста. Но есть и некоторое различие. Нажатие клавиши ESC приводит к возврату на меню более высокого уровня. Для выхода на ранее активное окно (редактора или сообщений) нужно нажать F6.

Приведем небольшой обзор, который должен помочь вам не «заблудиться» в интегрированной среде. Начнем слева направо основного меню.

Опция File (Работа с файлами)

Как следует из названия, опция предназначена для работы с редактируемыми файлами: с ее помощью

можно загружать, сохранять свои файлы, а также выполнять ряд других важных функций. На первых шагах освоения Турбо Си это будет для вас самая популярная опция. Поэтому работу с ней рассмотрим более подробно.

Нажатие на ENTER после выбора опции приводит к «выпаданию» подменю, каждый элемент которого соответствует определенному действию над файлом. Перечислим эти элементы (в последовательности сверху вниз).

Load (Загрузка) [вызывается из редактора клавишей F3] — загружает для редактирования файл с диска. В появившемся после нажатия на ENTER окне-подсказке можно указать имя нужного файла (можно указать имя несуществующего файла — после подтверждения будет создан и загружен в редактор новый файл), а можно оставить символы общности (* или ?). В последнем случае появляется окно со списком имеющихся в данном каталоге файлов с соответствующими именами. Пользуясь клавишами управления курсором, выберите нужный и нажмите на ENTER. Помимо списка файлов, в окне имеется изображение имен подкаталогов (большими буквами) и знак выхода в каталог более высокого уровня (..). Если понадобится файл из другого каталога, выберите соответствующую отметку и нажмите ENTER. Так можно путешествовать бесконечно (однако текущий каталог ОС при этом не меняется!). Отказ от загрузки файла производится клавишей ESC.

Pick (Выбор) [Alt-F3] — позволяет выбрать файл из списка тех файлов, с которыми вы ранее работали. Выбор производится обычным образом. Внизу списка имеется элемент Load File (загрузить файл) — выбрав его, переходим к работе в уже описанном режиме Load (Загрузка).

New (Новый файл) — используется для создания нового файла и загрузки его в редактор. По умолчанию этот файл имеет имя NONAME.C. Для задания своего оригинального имени можно использовать опции меню Write To и Save (смотри далее). Кроме того, новый файл можно создать с помощью опции Load, указав в ней имя несуществующего файла.

Save (Сохранение) [F2] — сохранение редактируемого файла, при этом редактирование будет продолжено. Учтите следующее: если попытаться сохранить файл под именем NONAME.C, система запросит новое имя. Во всех других случаях файл сохраняется под прежним именем.

Write to (Записать в ...) — выполняет сохранение файла под другим именем (как новый или на место существующего). В появившееся окошко следует написать имя (полное) файла. При этом, если не указывать расширение файла, он запишется как файл .C. Если же имя существующего файла указано, система спросит разрешение на уничтожение старой версии: File Exist, Overwrite? [Y/N]. Нажав Y, вы подтвердите свои намерения переписать файл.

Directory (Каталог) — опция позволяет выбрать файл из появившегося списка файлов, имеющихся в текущем каталоге. Опция во многом похожа на Load (см. выше), однако при этом выводится список по собственным символам общности (по умолчанию — *.*, хотя можно установить свою комбинацию).

Change Dir (Смена каталога) — позволяет установить новый текущий каталог для MS-DOS. Опция может пригодиться в следующем случае. Турбо Си обычно записывает вновь создаваемые файлы в текущий каталог MS-DOS. Если Турбо Си запушен не из своего рабочего каталога, но нужно исправить ошибку, выберите эту опцию и укажите в прямоугольнике-подсказке имя нового текущего каталога. Учтите, таким образом нельзя создать новый каталог — имя должно соответствовать уже имеющемуся на диске.

OS Shell (Среда ОС) — осуществляется временный выход в среду MS-DOS, после чего можно выполнять любые команды операционной системы (не стоит, однако, загружать в память резидентные программы!) — обратный возврат производится командой EXIT.

Quit (Завершение) [Alt-X] — выбор опции приводит к прекращению работы с Турбо Си.

Другие опции меню

Продолжим знакомство с основным меню. Следующая опция — **Edit (Редактирование)**. Выбор этой опции приводит к возврату в режим работы с текстовым редактором. Применительно к ней интерес представляют многочисленные команды редактирования, о которых мы расскажем в следующей главе.

Run (выполнение) предоставляет возможности для компиляции, компоновки и запуска разрабатываемой программы. Выбор опции приводит к «выпаданию» подменю, в котором первый элемент (Run), собственно, и выполняет указанные функции. Остальные элементы используются при отладке. Особое место занимает User Screen (экран пользователя), который позволяет временно переключиться на тот экран, куда выводит информацию программа.

Compile (компиляция) также содержит подменю с опциями, которые предназначены для компиляции и компоновки программ, в том числе и состоящих из нескольких файлов (проектов). Последняя опция в подменю — Get Info — позволяет получить информацию о состоянии Турбо-системы вообще и вашей программы в ней в частности.

Project (проект) — опция целиком предназначена для организации сложных программ: связывания в один проект несколько исходных и объектных файлов для создания законченных программ. При выборе опции появляется подменю, содержащее соответствующую операцию.

Options (опции) позволяет изменить различные параметры, влияющие на работу интегрированной среды. Некоторым вопросам использования этой опции мы посвятим следующий раздел.

Debug (отладка) и **Break/Watch (точки прерывания/отслеживание)** — эти две опции целиком посвящены предоставлению возможности отладки своей программы. С помощью имеющихся в них функций можно определять важные параметры для отладки, определять точки прерывания программы, отслеживать значения переменных и т.д. Отладка при этом проводится и с использованием функций подменю Run.

Установка параметров Турбо-среды — подменю Options

Корректность параметров обусловливает нормальные условия работы с системой. Причем параметры можно разделить на две категории. Первые влияют на удобства работы (например, использование табуляции, число строк на экране и так далее). Вторые фатально воздействуют на сам процесс работы — при их ошибочности разработка программ становится невозможной. Мы расскажем о том, как избежать такого развития событий и правильно установить параметры «второго вида». Выбор опции Options приводит к появлению подменю, элементы которого имеют следующее назначение.

COMPILER	—	установка параметров компилятора;
LINKER	—	установка параметров компоновщика;
ENVIROMENTS	—	установка параметров окружения;
DIRECTORIES	—	имена используемых Турбо Си каталогов;
ARGUMENTS	—	задание строки аргументов для вашей программы;
SAVE OPTIONS и RETRIEVE OPTIONS	—	сохранение новых параметров и восстановление старых соответственно.

Прежде всего особое внимание уделите установке правильных имен используемых Турбо Си каталогов. Для этого выберите элемент Directories и нажмите на ENTER. На экране в виде подменю появляется список каталогов. Теперь следует убедиться в том, что указанные имена соответствуют действительности. Далее перечислим каталоги:

Include Directories — каталоги включаемых файлов; нужно указать имена всех каталогов, содержащих те файлы, которые сейчас указываются в программе оператором #include <filename.h>, имена каталогов (если их несколько) следует разделять точкой с запятой, например:

C:/TC/INCLUDE;C:/WORK

Turbo C Directory — каталог, где будет искаться файл конфигурации и файл подсказки (TCHELP.TCH). В нашем примере следует указать:

C:/TC

Library Directories — указывается перечень каталогов, где находятся объектные (CO?.OBJ) и библиотечные (.LIB) файлы.

Pick File Name — указывает имя PICK-файла (файла списка) для системы Турбо Си.

Output directory — здесь требуется указать имя каталога, в который будут записываться вновь создаваемые файлы с расширениями .OBJ, .MAP и .EXE. Очевидно, это ваш рабочий каталог:

C:/WORK

Затем можно установить некоторые опции компилятора. Это прежде всего модель памяти. В Compiler выберите элемент Model и убедитесь, что там стоит **small (малая)**. Для начальных упражнений с Турбо Си рекомендуется именно такая память.

В подменю Compiler можно установить и ряд других опций. Например, определить длину значимых символов в идентификаторах программы — элемент **Indetifier Length**. Можно разрешить или запретить вложенные комментарии в программе — опция **Nested comments**. Мы уже говорили о некоторой несовместимости Турбо Си с ANSI стандартом. Запретить использование «доморощенных» ключевых слов Турбо Си можно опцией **ANSI Keywords**.

Пока ограничимся этими параметрами компилятора. Посмотрим, однако, что можно изменить для «более удобного» использования интегрированной среды. Для этой цели служит подменю, «выпадающее» после выбора **Enviroments**. Посмотрим, что можно делать с этим подменю (помните — многие опции «включаются» при установке напротив них метки **On** , выключаются — меткой **Off**, для смены меток используется клавиша ENTER).

Message Tracking — трассировка сообщений, подробно опция описывается при рассказе о процессе отладки программы;

Keep Messages — сохранение сообщений, при установленной в On опция вызывает сохранение сообщений об ошибках и замечании от ранее проведенных компиляций программы;

Config Auto Save — при выключении данной опции конфигурация сохраняется только по команде SAVE OPTIONS в подменю OPTIONS; установленная в On, опция автоматически сохраняет измененную конфигурацию Турбо-системы;

Edit Auto Save — автоматическое сохранение (On) редактируемого файла при завершении работы с Турбо;

Backup files — при установке в On вызывает создание .BAK копии файла при сохранении последнего командой SAVE;

Tab size — шаг табуляции, общепринято восемь позиций, а вообще можно и больше;

Zoomed window — «распахивание» текущего окна (как и F5);

Screen size — можно установить число линий на экране компьютера при работе с Турбо Си — для терминалов типа EGA возможно либо 25, либо 43, а для VGA — 25 или 50 линий.

Существует еще целый набор интересных и важных опций, но их рассмотрение пока отложим.

Строка быстрой подсказки и«горячие клавиши» Турбо Си

В низу экрана находится уже упоминавшаяся строка быстрой подсказки, в которой перечисляются так называемые «горячие клавиши». Нажатие на данные клавиши немедленно приведет к выполнению соответствующий операции. Строка содержит список тех клавиш, которые задействованы для текущего режима работы Турбо Си. Обратите внимание — при работе в редакторе и использовании основного меню список клавиш не меняется. Однако, вызвав подсказку (по клавише F1), можно получить совершенно новую строку. Учтите, что мы рассматриваем клавиши применительно к **Турбо Си 2.0**, в более ранних версиях имеются некоторые различия.

Рассмотрим наиболее «популярный» вариант строки подсказки, устанавливаемый при работе в редакторе и в основном меню. По причине ограниченности длины строки экра-

на в подсказке перечислены только основные клавиши. Обратите внимание — при нажатии на клавишу Alt строка подсказки меняется и показывает набор команд, вводимых с помощью комбинации Alt-некая клавиша. Строка имеет следующий вид:

```
F1-Help F5-Zoom F6-Message F7-Step
F8-Trace F9-Make F10-Main menu
```

F1-Help — вызывает систему подсказки Турбо Си;

F5-Zoom — увеличивает размеры активного окна до размеров всего экрана;

F6-Message — переключает активные окна — окно сообщений на окно редактирования и обратно;

F7-Trace — вызывает компиляцию с последующим пошаговым выполнением программы, при этом пошагово выполняются и функции (подробнее см. далее);

F8-Trace — аналогично Trace, однако при пошаговом выполнении функции выполняются за один шаг;

F9-Make — производит компиляцию программы и создает EXE-файл.

F10-Main menu — Осуществляет выход из редактора в основное меню.

Активны такие важные клавиши, «не уместившиеся» в строке подсказки, как:

F2 — сохраняет файл, находящийся в редакторе;

F3 — загружает файл в редактор;

F4 — вызывает пошаговое выполнение программы со строки, на которой установлен курсор (естественно, предыдущие операторы также выполняются);

CTRL-F1 — вызов подсказки на данное ключевое слово;

CTRL-F2 — прерывает пошаговое выполнение программы;

CTRL-F3 — показывает содержимое стека;

CTRL-F4 — вызывает режим оценки значения переменных;

CTRL-F7 — включает/выключает точку прерывания выполнения прогаммы;

CTRL-F8 — добавляет так называемый watch (отслеживание) в программу;

CTRL-F9 — компилирует редактируемый файл, создает файл объектного модуля с расширением .OBJ и запускает его на выполнение.

Мы уже говорили, что при нажатии на Alt появляется"альтернативная" подсказка. В ней перечислены следующие команды:

```
Alt: F1-Last Help F3-Pick F6-Swap
F7/F8-Prev/Next Error X-Exit;
```

ALT-F1 — вызывает последний экран системы помощи;

ALT-F3 — позволяет осуществить выбор файла для загрузки;

ALT-F5 — производит переключение между пользовательским экраном и экраном Турбо Си;

ALT-F6 — загружает в редактор последний редактировавшийся файл;

ALT-F7/F8 — переход к предыдущей/следующей ошибке (см. далее);

ALT-F9 — компилирует файл в редакторе и создает файл объектного модуля с расширением .OBJ;

ALT-X — Осуществляет выход из среды Турбо Си;

Кроме того, напомним, что нажатие клавиши Alt и первой буквы имени опции основного меню активизирует данную опцию. Например, ALT-F задействует опцию File. Наконец, нажатие Shift-F10 ознакомит с авторскими правами на Турбо Си и версией данного продукта.

ПРИМЕЧАНИЕ. Мы упоминули некоторое различие, которое имеется со списком «горячих ключей», реализованных в предыдущих версиях. Наиболее существенное для пользователя (на первое время) — это отсутствие многих опций отладки (отсутствие подменю в опции RUN и отсутствие самой опции Break/Watch) и соответствующих клавиш. Еще более существенно различие в клавишах F7 и F8. В версии 2.0 это пошаговое выполнение программы, в версии 1.0 — переход к предыдущей/следующей ошибке (в версии 2.0 эту роль выполняют клавиши ALT-F7/F8 соответственно). Так что, прочитав статью, поэкспериментируйте с клавишами: практика — лучший способ освоения новых систем.

ТЕКСТОВЫЙ РЕДАКТОР ТУРБО СИ

Работа редактора Турбо Си напоминает работу известного текстового редактора «WordStar», а также редакторов системы «SideKick» и других продуктов фирмы «Борланд». Если Вы ранее работали с каким-либо

из названных редакторов, то можно просто просмотреть данный раздел и переходить к следующему.

Редактор запускается после выбора опции EDIT в основном меню (для обратного выхода из редактора в меню следует нажать функциональную клавишу F10). При этом в окне редактора появляется курсор, и можно заняться соответствующей работой с текстом.

Когда запускается Турбо Си, то он создает файл с именем NONAME.C или загружает файл, указанный в конфигурации. Сначала вас могут устроить такие действия. Однако впоследствии появляется необходимость редактирования уже имеющихся файлов и сохранения внесенных изменений на диске. Для этих целей используйте опцию File основного меню (или некоторые функциональные клавиши). Операции над файлами мы уже рассмотрели при рассказе об опции File.

Строка состояния редактора и команды редактора

Вверху окна редактора находится подсвеченная (бледная) строка, которая называется строкой состояния редактора и содержит некоторую информацию о редакторе и редактируемом файле. Строка состояния имеет вид:

```
Line Col Insert Indent
Tab C:FILENAME.C
```

Элементы несут следующую полезную информацию:

Line — номер строки, в которой находится курсор;

Col — номер столбца, в котором находится курсор;

Insert — указывает, активизирован (On) или нет (Off) режим вставки. Обратный режим — режим затирания или замены. Переключение режимов проводится с помощью клавиши INS или CTRL-V;

Indent — активизировано или нет смещение (левый отступ) начала строки текста или нет (соответственно On и Off). Включение и выключение отступа осуществляется нажатием комбинации CTRL-O I;

Tab — указывает, что режим табуляции включен. Переключение режима табуляции достигается нажатием комбинации CTRL-O T.

Последний элемент строки состояния указывает дисковод, имя и расширение, имя редактируемого файла.

Теперь остановимся на том, как вводить команды редактора. За некоторым исключением все команды редактора вводятся комбинацией управляющей клавиши CTRL и какой-то другой клавиши. За этим следует нажатие еще одной клавиши. Например, комбинацию CTRL-Q F (поиск строки) следует набирать в такой последовательности: сначала нажать CTRL-Q и затем F (строчную или прописную).

Перемещение курсора и операции удаления

Перемещение курсора по редактируемому тексту осуществляется традиционными клавишами управления курсором. На всякий случай мы приводим их в таблице. Обратите внимание на то, что команды управления курсором можно вводить и нажатием двух клавиш (CTRL и определенной буквы). Удаление символов так же производится клавишами DEL и BACKSPACE.

Но существуют и характерные для Турбо Си команды. Удалить целиком слово справа от курсора можно, нажав комбинацию CTRL-T. (При этом словом считается последовательность символов, которая отделена от других символов одним из следующих разделителей: пробел $ / — + * ' ^ [] (. : , < >).

Всю строку можно удалить нажатием комбинации CTRL-Y. Если необходимо удалить весь текст от позиции курсора до конца строки, нажмите комбинацию CTRL-Q Y. Перечень команд удаления приведен в таблице.

Блоки текста и работа с ними

Редактор Турбо Си позволяет манипулировать целыми блоками текста, т.е. пересылать или копировать их в другое место или вообще удалять. Все эти действия требуют, чтобы сначала было выделено начало блока (CTRL-K B), потом конец блока (CTRL-K K), а затем введена команда о том, что с блоком делать.

Для перемещения блока необходимо переместить курсор в новое место назначения и нажать CTRL-K V. Для копирования блока вместо CTRL-K V нажмите CTRL-K C. Для удаления отмеченного блока нажмите комбинацию CTRL-K Y. В тексте можно выделить и одно-единственное слово, поместив курсор на первый символ слова и нажав комбинацию CTRL-K T.

В Турбо Си имеется возможность перемещать текстовые блоки на дисковый файл для последующего использования. Это можно сделать путем выделения требуемого блока и нажатия комбинации CTRL-K W. После этого Турбо Си просит указать имя файла, в котором будет храниться блок. Исходный блок в тексте программы при этом не удаляется. Для считывания блока в память нажмите CTRL-K R (Турбо Си также запросит имя файла).

Перечень команд редактора Турбо Си по категориям

Команда	Действие
= ПО ЦЕНТРУ	TABLE TEXT
Команды курсора (специфичные)	
= TABLE TEXT	ПЕТИТ
CTRL-A	Перемещает курсор на одно слово влево
CTRL-F	Перемещает курсор на одно слово вправо
CTRL-W	Прокручивает экран вверх
CTRL-Z	Прокручивает экран вниз
CTRL-Q E	Перемещает курсор в верх экрана
CTRL-Q X	Перемещает курсор в низ экрана
CTRL-Q R	Перемещает курсор в начало файла
CTRL-Q C	Перемещает курсор в конец файла
CTRL-Q B	Перемещает курсор в начало блока
CTRL-Q K	Перемещает курсор в конец блока
CTRL-Q P	Перемещает курсор в предыдущую позицию

Команды вставки

TNS или CTRL-V	Переключает режим вставки
ENTER или CTRL-N	Вводит пустую строку

Команды удаления

CTRL-Y	Удаляет всю строку
CTRL-Q Y	Удаляет символы от курсора до конца строки
BACKSPACE	Удаляет символ слева от курсора
DEL или CTRL-G	Удаляет символ в позиции курсора
CTRL-T	Удаляет слово справа от курсора

Блочные команды

CTRL-K B	Отмечает начало блока
CTRL-K K	Отмечает конец блока
CTRL-K T	Отмечает слово
CTRL-K C	Копирует блок
CTRL-K Y	Удаляет блок
CTRL-K H	Скрывает или проявляет блок
CTRL-K V	Перемещает блок
CTRL-K W	Записывает блок на диск
CTRL-K R	Считывает блок с диска

Команды поиска

CTRL-Q F	Осуществляет поиск
CTRL-Q A	Осуществляет поиск с заменой
CTRL-Q N	Поиск маркера
CTRL-L	Повторяет поиск

Дополнительные команды

CTRL-V	Отменяет вызванную команду
CTRL-O I	Включает и выключает режим автоматического левого отступа
CTRL-P	Дает возможность вводить управляющий символ
F10	Осуществляет выход из редактора

F3	Позволяет приступить к созданию нового файла
CTRL-Q W	Восстанавливает затертое сообщение об ошибке
F2	Сохраняет файл
CTRL-K N	Устанавливает маркер блока
CTRL-O T	Включает и выключает режим табуляции
CTRL-O L	Отменяет результат выполнения последней команды

Поиск и поиск с заменой

Ввод в режим поиска производится нажатием CTRL-Q F. Турбо Си в строке состояния требует ввести нужную строку символов. Введите строку и нажмите ENTER. После этого Турбо Си в строке состояния запросит параметры поиска. Существует ряд параметров, определяющих режим поиска (задавать их не обязательно, можно просто нажать клавишу ENTER):

Параметр	Действие
B —	Просмотр осуществляется вверх по файлу от текущей позиции курсора;
G —	Просматривается весь файл независимо от положения курсора;
N —	Определяет режим замены — автоматический (по умолчанию) или после подтверждения;
U —	Ищутся только заглавные или строчные буквы;
W —	Ищутся только целые слова, подстрочные игнорируются;
целое число —	Ищется n-е появление строки.

Поиск можно повторить, нажав CTRL-L. Для задействования поиска с заменой нажмите CTRL-Q A. Работа этой команды аналогична простому поиску, только найденная строка заменяется другой. Так, если указан параметр N, Турбо Си не будет спрашивать, заменять или нет найденную строку, а сделает это автоматически.

Установка и поиск маркеров. В один файл можно поместить до четырех маркеров, отмечающих определенные позиции в тексте. Для ввода маркера нажмите CTRL-K и введите номер маркера n (от 0 до 3). Потом можно вернуться на отмеченную маркером позицию, введя CTRL-Q n (где n — номер маркера).

ВАША ПЕРВАЯ ПРОГРАММА НА ТУРБО СИ

Мы рассмотрели различные опции меню Турбо Си и возможности редактирования текстовых файлов. Однако, очевидно, остается открытым основной вопрос: каким же образом можно разрабатывать Си-программы, используя описанные выше возможности.

Рассмотрим теперь такие темы: разработка и компиляция простой ("первой") Си-программы, разработка проектов и отладка программы с помощью встроенных возможностей Турбо Си, компиляция и компоновка в среде интегрированной системы.

Создание новой программы в среде интегрированной системы Турбо Си обычно включает в себя, как отмечено в Руководстве, шесть шагов: от запуска Турбо Си до запуска программы. Расскажем об этом интересном процессе, соблюдая подобную традицию.

Шаг 1. Запуск системы Турбо Си

Первый шаг уже был рассмотрен. Повторимся: наберите на приглашение операционной системы ТС или имя разработанного вами командного файла.

Если требуется произвести запуск прямым указанием имени файла TC.EXE, учтите следующее: интегрированная система Турбо Си может принимать два аргумента от командной строки. Это имя файла, который должен быть загружен в редактор, и ключ /C, за которым следует еще одно имя файла без пробела. Эти два аргумента могут следовать в любом порядке. Например,

```
c hello /cmyconfig
```

поместит HELLO.C в редактор и загрузит файл конфигурации MYCONFIG.TC. Заметьте, между ключом /C и именем файла отсутствует пробел, а также допустимы (по умолчанию) расширения: .C для редактируемого файла и .TC для файла конфигурации.

Шаг 2. Поиск каталога с вашей программой

После «старта» нужно определиться: редактировать уже имеющуюся на диске программу или создавать новую. Если надо взять имеющуюся, Руководство советует вторым шагом найти нужный каталог и сделать его текущим. Для установки каталога текущим следует выбрать опцию File основного меню (мы рассказывали, как это делать — нажать Alt-F). В подменю выбрать CHANGE DIR и указать новое имя.

Необходимость установить новый текущий каталог возникает в том случае, когда требуется работать в своем собственном каталоге, но запускали Турбо Си из другого.

Шаг 3. Проверка параметров рабочего окружения

Еще раз выберите в подменю опции Options элемент DIRECTORIES и проверьте соответствие указанных там каталогов реальной конфигурации пакета Турбо Си. При описании основного меню мы уже сказали о порядке установки параметров.

Шаг 4. Загрузка программы или создание ее редактором

На шаге 4 производится загрузка программы в редактор или создание заново исходного модуля. Нажмите F3 (считать файл) для загрузки программы с диска. Затем можете явно ввести имя программы HELLO.C — если работа со своим примером начинается впервые, после соответствующего запроса нужно создать новый файл. Так или иначе, в окне текстового редактора будет следующий текст:

```
#include .h
main ()
{
  printf("Hello world/n");
}
```

Шаг 5. Создание выполняемого файла

На этом шаге будем использовать меню опции Compile. Откройте его (Alt-C) и посмотрите — здесь напротив опций Compile to OBJ (компилировать в объектный код) и Link EXE file (компоновка файла) будет указано имя транслируемого файла и имя создаваемого выполняемого файла .EXE. В нашем простом варианте однофайловой программы этим файлам также будет присвоено имя HELLO, что очевидно. Однако при работе с программами, состоящими из нескольких файлов, дела обстоят сложнее!

Нужно откомпилировать исходный файл и выполнить компоновку получившегося объектного файла со стандартными библиотеками и файлом загрузки. Это можно сделать различными способами. Можно выбрать опцию Compile to OBJ (или нажать Alt-F9) а затем Link EXE file.

Но легче всего выбрать опцию MAKE EXE FILE (создать файл EXE) или нажать F9. Завершением всех этих действий будет сообщение об успешной компиляции и компоновке.

Шаг 6. Последний?

Завершает наш процесс запуск программы. Для этого нажмите Ctrl-F9 или выберите Run (выполнение) в подменю опции Run основного меню.

Если все прошло нормально, на экране что-то мелькнет, после чего все возвращается в исходное положение. Очевидно, результаты работы программы остались «под экраном» — для доступа к ним нажмите Alt-F5 (опция User Screen — экран пользователя, — в подменю Run). Надеемся, что вы обнаружите после этого классическое «HELLO, WORLD!»

Ошибки и сообщения

Не все может быть так успешно и красиво. Часто все выглядит обратно — в программе оказывается огром-

ное количество ошибок, сообщения о которых наводняют соответствующее окно. Выявить и исправить ошибки — важная и не очень приятная задача программиста.

Рассмотрим, как поступает Турбо Си с сообщениями об ошибках и предупреждениях. Турбо Си сначала записывает все сообщения компилятора и компоновщика в буфер, а затем выводит их сразу в окно сообщений. Это позволяет ознакомиться с ними в условиях, когда сохраняется возможность непосредственно редактировать тексты исходных файлов.

Для проверки этой возможности добавьте несколько синтаксических ошибок в пример программы. Уберите знак # из оператора включения в первой строке. Затем уберите последние кавычки в пятой строке программы — printf. Получившийся файл с ошибками будет выглядеть следующим образом:

```
include <stdio.h>
main()
{
printf("Hello world/n);
```

Теперь откомпилируйте файл снова, нажав Alt-F9. Окно компиляции подскажет, сколько получилось в результате ошибок и предупреждений (должно получиться три сообщения об ошибках и ни одного предупреждения).

При появлении в окне компиляции сообщения Press any key (нажмите любую клавишу) нажмите пробел. На экране появится окно сообщений, и светлая полоса (маркер-строка) расположится на первой ошибке или предупреждении. Поскольку первая ошибка относится к файлу, находящемуся в данный момент в окне редактора, то появится подсвеченная строка, отмечающая то место в программе, где компилятор нашел что-то, напоминающее ошибку.

Если нужно просмотреть другие сообщения, то воспользуйтесь клавишами управления курсором для перемещения маркер-строки в окне сообщений вверх и вниз. Обратите внимание, что эта строка перемещается как в окне редактора, так и в окне сообщений по компиляторам и ошибкам. При желании посмотреть строку, на которой вы находились перед выходом из редактора, поместите маркер-строку на сообщение о компиляции.

Если текст в окне сообщений имеет слишком большую длину и не умещается в окне, то можно использовать клавиши со стрелками влево и вправо для горизонтального продвижения сообщения. Для обзора большего числа сообщений одновременно можно увеличить окно сообщений до размеров всего экрана, нажав F5 (Zoom). Увеличившееся окно сообщений может заслонять окно редактора и слежение не выполняется, тогда снова верните окна в режим наложения.

Исправление синтаксических ошибок

Для исправления ошибки поместите маркер-строку окна сообщений на сообщение об ошибке и нажмите ENTER. Маркер перекинется в окно редактора и расположится на месте, которое вызвало сообщение об ошибке. Заметьте, что строка состояния редактора показывает выбранное сообщение (это полезно при работе с распахнутым окном). Теперь можно исправить ошибку, которая породила сообщение (для этого в нашем примере поставьте знак #).

Поскольку данное сообщение не единственное, можно вернуться в окно сообщений, нажав F6 (Message) и выбрать очередное сообщение так, как это было сделано ранее.

Однако не обязательно возвращаться в окно сообщений для того, чтобы перейти к следующей ошибке. Можно просто нажать ALT-F8 (Next error — следующая ошибка, в версии 1.0 — клавишу F8), и редактор поместит курсор на место, соответствующее очередной ошибке окна сообщений. Заметьте, что сообщение, показанное в строке состояния и в высвеченной строке окна сообщений, меняется по мере того, как вы продвигаетесь от одной ошибки к другой. По аналогии можно двигаться и в обратном направлении, нажав Alt-F7 (Previos error — предыдущая ошибка, в версии 1.0 — клавиша F7!).

Имеются определенные преимущества у обоих этих методов. Иногда одна глупая ошибка в исходной программе может «смутить» компиля-

тор, вызвав множество сообщений. Исправление такой ошибки делает все остальные сообщения бессмысленными, и тогда наиболее удобно использовать первый метод — вернуться к окну сообщений после исправления первой ошибки, переместиться вниз к следующему значащему сообщению и выбрать его. В других случаях имеет смысл проверить все сообщения последовательно; нажатие Alt-F8 (следующая ошибка) в этом случае более эффективно.

Запомните, что Alt-F7/F8 являются клавишами оперативного доступа, т.е. работают из любого места в среде программирования Турбо Си. Так, если вы находитесь в окне сообщений и нажмете F8 (следующая ошибка), то не получите сообщения, которое высвечивается текущим, но получите следующее за ним сообщение. (Если требуется вызвать текущее сообщение, нажмите Ввод.) Если больше у компилятора нет сообщений, то Alt-F8 не даст никакого результата. Замечание: сообщения компоновщика не могут быть выбраны и не отслеживаются в вашем исходном файле.

Построение проектов

Наконец мы добрались до разработки Си-программы из нескольких раздельно компилируемых исходных файлов. Для этого необходимо точно сообщить Турбо Си, какие файлы используются в программе. Сделать это можно с помощью файла проекта.

Создать файл проекта более чем просто. Рассмотрим процесс на небольшом примере. Разработаем несколько файлов. Лучше всего создать файл с главной программой и вспомогательный файл с определениями функций и данных, вызываемых из главной программы. В примере из Руководства пользователя главный файл MYMAIN.C выглядит таким образом:

```
#include .h
char *GetString (void);
main(int argc, char *argv[])
{
/* При запуске программы указывается чье-то имя... */
    char *s
    if (argc  1)
        s = argv[1];
    else
        s = «Незнакомец»;
    printf("%s %s./n",GetString(),s);
}
```

Вспомогательный файл MYFUNCS.C для примера выглядит так:

```
char ss [] = «Добрый день, уважаемый »;
char *GetString(void)
{
    return ss;
}
```

Теперь подходим к файлу проекта. Это обычный текстовый файл с расширением .PRJ, в котором содержится несколько строк (в нашем случае — две) с названиями файлов, которые нужно откомпилировать и скомпоновать. Назовем файл проекта MYPROC.PRJ. Он должен выглядеть так:

```
mymain
myfuncs
```

Можно указывать расширения файлов, можно не указывать, можно указывать пути к файлу (Турбо Си станет искать его в указанном каталоге), можно не указывать (файл будет отыскиваться в «известных» Турбо Си каталогах). Наконец, порядок файлов в файле проекта не имеет никакого значения.

Заметим, что имя файла проекта (MYPROG.PRJ) может отличаться от имени главного файла (MYMAIN.C). Однако имена (но не расширения) могут быть и одинаковыми. Важно запомнить, что имя выполняемого файла будет определяться именем файла проекта. В нашем случае выполняемым файлом будет MYPROG.EXE.

Можно указать маршрут к любому файлу, перечисленному в файле проекта. Таким образом, можно создавать программу не только из файлов, расположенных в одном и том же каталоге.

Для работы с проектами следует использовать функции, реализуемые опциями **Project** и **Compile** основного меню. Перед компоновкой проекта следует в подменю первой опции найти пункт **Project name** (имя проекта), нажать на ENTER и указать имя файла-проекта (можно явно ввести имя своего файла или воспользоваться символами общности (? и *) для на-

хождения его в каталоге). Когда имя проекта введено, можно приступать к компиляции и компоновке соответствующих файлов. Можно использовать две функции подменю Compile:

Make EXE file (создать файл EXE) вызывается по клавише F9 — вызывает процедуру MAKE-PROJECT системы Турбо Си, которая проводит трансляцию (тех файлов, для которых нет модулей .OBJ) и компоновку файлов;

Build All (все переделать) — аналогично первой, но заново перекомпилирует все файлы.

Для запуска необходимо выбрать в меню Run соответствующую команду (тоже Run) — CTRL-F9.

Заметим, что запуск программы включает в себя и ее создание (т.е. вызов MAKE-PROJECT). Таким образом, можно «опустить» компиляцию и сразу нажимать CTRL-F9.

Отслеживание ошибок в проекте

В примере с единственным исходным файлом показано, что синтаксические ошибки могут быть просмотрены и выбраны из окна сообщений. Аналогично окно сообщений позволяет обрабатывать ошибки при многофайловой компиляции.

Для того чтобы это увидеть, нужно ввести некоторые синтаксические ошибки в оба файла: MYMAIN.C и MYFUNCS.C. Из MYMAIN.C в первой строке удалить первую острую скобку, а в пятой — удалить C в **char**. Эти изменения должны генерировать три ошибки и три предупреждения в MYMAIN.

Теперь загрузите MYFUNCS.C и удалите первое **r** из **return** в пятой строке. Это изменение вызовет две ошибки и два предупреждения.

В подменю Project есть опция **Break make On** (прервать создание). Установленная в ERROR, она говорит о необходимости остановиться при появлении ошибок. Установите опцию в данное положение.

Итак, нажмите клавишу F9. Когда появится информация об ошибках и фраза Press any key, нажмите пробел.

Ваш курсор должен располагаться на первой ошибке или предупреждении в окне сообщений. Если файл, к которому относится это сообщение, находится в редакторе, яркая подсветка в окне редактора покажет место ошибки. Далее можно перемещаться вверх и вниз в окне сообщений для просмотра различных сообщений. Когда вы подойдете вниз к границе файла, окно редактора может перейти или не перейти на следующий. Это завист от состояния **Message tracing** (отлеживание сообщений) в опции Enviroments подменю Options.

По умолчанию отслеживание сообщений производится только в текущем файле (установлено **Current file**). Итак, перемещаясь к сообщению, которое относится к файлу и которого нет в окне редактора, подсветка выключается. Если выбирается одно из этих сообщений (т.е. нажимается Enter, находясь на сообщении), соответствующий файл будет загружен в редактор и курсор укажет место ошибки.

Однако, установив переключатель в режим **All files** (все файлы), можно обрабатывать сообщения через границы файлов. Турбо Си автоматически будет загружать соответствующий файл в редактор. Можно также полностью отключить обработку ошибок, установив переключатель Message tracing в режим **OFF**. В этом случае все просто — выбираем сообщение, с которым будем работать, и нажимаем ввод. Соответствующий файл загружается в редактор, а курсор указывает на место ошибки.

Клавиши Alt-F7 и Alt-F8 (предыдущая ошибка и следующая ошибка) не действуют на изменение режимов переключателя Track messages. Эти полезные клавиши всегда будут находить следующую или предыдущую ошибку и загружать соответствующий файл, если это необходимо.

Отметим еще два пункта подменю Project. Сбросить проект (т.е. стереть его имя) можно с помощью **Clear Project** (очистка проекта). Выбор пункта **Remove messages** (удаление сообщений) приведет к очистке информации, содержащейся в окне сообщений.

Использование опций Debug (отладка) и Break/Watch (прерывание/наблюдение)

Интегрированная среда Турбо Си предоставляет многочисленные возможности по отладке программы и поиску в ней всевозможных ляпсусов (а также, очевидно, и более детального понимания работы самой программы). Рассмотрим только некоторые из них.

Расскажем о пошаговом выполнении программы, которое позволяет последовательно пройти всю программу от начала и до конца пооператорно, т.е. «шаг за шагом». Для этого нужно использовать пункты меню RUN Trace Into (клавиша F7) и Step over (F8). Выбор первого приводит к тому, что пошагово будут выполняться все функции, имеющиеся в программе (Турбо Си «заглядывает» внутрь функций). Вторая опция приводит к «поверхностному» выполнению основной процедуры main(), при этом имеющиеся в ней функции выполняются мгновенно.

Нажатие клавиш F7 и F8 первым делом приводит к попытке скомпилировать и скомпоновать вашу программу. Если при этом обнаружились ошибки, никакого пошагового выполнения не будет — этот режим используется для верификации уже формально правильных программ!

После успешной компиляции в начале main() появляется голубой курсор, отсвечивающий соответствующую строку. Далее снова следует нажать F7 (F8) — отмеченный курсором оператор выполнится, курсор передвинется далее. И так до конца программы.

Что можно делать во время этих шагов? Можно следить за экраном пользователя (User Screen), нажимая после выполнения шага Alt-F5. С помощью опции подменю Debug Evaluate (оценить), вызываемой также нажатием Ctrl-F4, можно отслеживать значения всех переменных в программе: после вызова данной опции появляется экран из трех окошек — Evaluate (где указывается имя переменной), Result (значение), New result (новое значение — его можно установить самостоятельно, повлияв на значения переменной). Завершение работы с режимом оценки — ESC.

Можно вызвать режим просмотра стека (клавиши Ctrl-F3 или Call Stack в подменю Debug). В правом верхнем углу экрана появится содержимое стека — функции, которые в данный момент активны (внизу стека всегда main()). Выбрав имя функции и нажав ENTER, курсор редактора установится на данную функцию.

Еще две возможности. Сбросить пошаговый режим можно, не доходя до последнего оператора. Для этого служит Restart в подменю Compiler (или клавиши Ctrl-F2). Можно начать пошаговое выполнение с некоторого места в программе — установите на это место курсор и выберите Goto cursor в подменю Compiler (или нажмите F4).

ПОКА ВСЕ!

Другие опции Турбо Си позволяют применять еще более хитрые методы для отладки программ. Огромное число функций в библиотеках Турбо Си позволяют быстро и эффективно решать самые разнообразные задачи. Турбо Си поддерживает работу с графикой — только этому посвящено немало толстых книг (к сожалению, вышедших за рубежом). Словом, тему Турбо Си можно продолжать бесконечно долго. Но мы и так рассмотрели достаточно много вопросов для первого знакомства. Пожалуй, стоит ограничиться этим и попрактиковаться в практическом программировании на Турбо Си. И почаще нажимайте кнопку F1 — общение с системой подсказки частично заменит вам отсутствие хороших и доступных книг. Мы, в свою очередь, надеемся продолжить разговор на данную тему. А пока желаем вам успешных первых программ!

ЛИТЕРАТУРА ПО ТУРБО СИ

1. Керниган Б., Ритчи Д. Язык программирования Си.- М.: Финансы и статистика, 1985.

2. Turbo C 2.0, User's Guide, Borland Int., Scotts Valley USA, 1988.

3. Turbo C 2.0, Reference Manual, Borland Int., Scotts Valley USA, 1988.

♥ ♥ ♥

О 38 Операционная система UNIX. - М.: Знание, 1991. - 48 с. - (Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение"; N 6).
ISBN 5-07-001720-9
35 к.

Операционная система UNIX проникает сейчас в компьютеры всех классов. Но разработка ее еще далеко не завершена, еще не решены некоторые стандартные проблемы, которые и обсуждены в этом выпуске.
Рассчитан на широкий круг читателей.

2302030000 **ББК 32.85**

ТЕМА *СЛЕДУЮЩЕГО* **НОМЕРА:**

РАДИОЭЛЕКТРОНИКА И СВЯЗЬ

ОПТОЭЛЕКТРОННЫЕ СИСТЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

ПАКЕТ PC TOOLS

МАТЕМАТИКА КИБЕРНЕТИКА

ОПЫТ ГРУППОВОГО АНАЛИЗА

Научно-популярное издание

ОПЕРАЦИОННАЯ СИСТЕМА UNIX

Гл.отраслевой редактор *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл.редактор *И. А. Васильева*
Художник *В. И. Конюхов*
Худож.редактор *И. А. Емельянова*
Техн.редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ N 11303

Подписано к печати 30.04.91. Формат бумаги 70x100[1]/16. Бумага офсетная. Печать офсетная. Усл.печ.л.3,90. Усл.кр.-отт.8,45. Уч.-изд.л.4,16. Тираж 49987 экз. Заказ 2167. Цена 35 коп. Издательство "Знание". 101835, ГСП, Москва, Центр, проезд Серова,д.4. Индекс заказа 914706. Отпечатано с оригинал-макета издательства "Знание" на ордена Трудового Красного Знамени Тверском полиграфическом комбинате Государственного комитета СССР по печати. 170024, г.Тверь, пр.Ленина,5.

Цена 35 коп.

Индекс 70195

Адрес подписчика:

СОШ 5-27

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Все формы похожи и ни одна не одинакова с другой; и так весь хор их указывает на тайный закон...

И.В.Гете

Геометрия приближает нас к истине

Платон

Никто не должен бояться, что наблюдение над знаками уведет нас от вещей: напротив, оно приводит нас к сущности вещей.

Лейбниц

Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4